# CommuniGate® Pro

## Internet Communications Server
### Email ~ Collaboration ~ VoIP

**CommuniGate Pro v. 6.2**

# CommuniGate Pro

**Introduction**
Installation
SysAdmin
Network
Objects
Storage
E-mail
Real-Time
Access
Services
Directory
Clusters
Applications
Miscellaneous
Licensing

WebMail
Pronto!
PBX

| Introduction | Features | History | How To | Help Me |
| --- | --- | --- | --- | --- |

## CommuniGate® Pro

- **CommuniGate Pro Product Description**
    - **Features**
    - **Administration**
- **Support and Discussions**
- **Updates and Bug Fixes**
- **Download the Latest Versions**
- **Download the CommuniGate Pro E-mail Plugins**
- **Download the CommuniGate Pro MAPI Connector**
- **Download the CommuniGate Media Plugin for Browsers**

Welcome to CommuniGate Pro, the Unified Internet Communication Server.

Based on open standards, CommuniGate Pro provides an integrated platform for Store-and-Forward (E-mail, Calendaring) and Real-Time (VoIP, Video, Instant Messaging, White Boards) communications over IPv4 and IPv6 networks.

CommuniGate Pro can be used via its built-in WebUser Interface, the bundled Pronto! communication client, as well as any third-party client applications employing the SMTP, IMAP, POP, MAPI, SIP, XMPP, HTTP, FTP, WebDAV, CalDAV, CardDAV, XIMSS, and other standard protocols.

CommuniGate Pro can exchange E-mail and Groupware information with other standard-based servers and systems using the SMTP protocol. CommuniGate Pro supports Real-Time communications (VoIP, IM, Presence) with other standard-based systems using the SIP and XMPP protocols.

# CommuniGate Pro Product Description

The main CommuniGate Pro subsystems include:

## Identity Management

- Multi-Domain architecture (field-proven for over 120,000 domains per system), with multihoming and shared-IP configurations.
- Account concept, including Mailbox Storage, File Storage, Account Settings, and Account Information databases.
- Groups, Forwarders, Aliases, and other Domain Objects.
- Meta-Directory with Local and Remote Units.
- LDAP access to Directory and Account databases.
- External Authentication mechanism for integration with 3rd party solutions.
- RADIUS services.
- Billing system with multiple per-Account Balances and reservations.

# Storage Management

- Mailbox Storage with multiple Mailboxes, Shared access, ACLs.
- Mailbox formats - text files, file folders, other data containers.
- File Storage with public and private folders, virtual files.
- Groupware Information storage and processing using the iCalendar and vCard standards.

# Mail Transfer

- Unified Message Queue processing.
- ESMTP and LMTP mail exchange services.
- Anti-Spam and other protection mechanisms.
- Plugin Interface for high performance virus, spam, and content filtering.
- Automated Mail Processing Rules.
- Mailing List manager with automatic bounce processing and a Web interface to list archives.
- Remote Account Polling using the POP3 protocol.
- External Program Delivery for custom applications.
- Automated Invitation processing for shared resource scheduling.

# Real-Time Signaling

- Unified Signal processing.
- XIMSS protocol for Instant Messaging, Presence, audio and video communications.
- SIP protocol for Instant Messaging, Presence, audio and video communications, desktop sharing and real-time collaboration.
- XMPP protocol for Instant Messaging and Presence.
- SMPP protocol for Instant Messaging via SMS.
- NAT Traversal mechanisms (near-end and far-end) for XIMSS, SIP, RTP, and TCP-based media protocols.
- Registrar, forking Proxy, and Presence server functionality.
- Automated Signal Processing Rules.
- Event packages for presence, message-waiting, registration, dialogs, and other services.
- STUN server for client-side NAT Traversal solutions.
- Parlay X Interface.

# Real-Time Application Environment

- Domain-specific application environments.
- Media Server component for call termination and conferencing with the built-in NAT traversal and encryption support.
- CG/PL language for quick and robust application design.
- Built-in operations for call control, bridging, and multi-party conferencing.
- Integrated Access to Message and File stores.

# Data Access Services

- Multi-protocol, multi-client simultaneous access to all Account data.
- POP3 and IMAP4 mail client access.
- MAPI interface for Microsoft® Windows clients (Outlook and other MAPI-enabled applications).
- WebUser Interface to Mailbox, Groupware and File Stores, to Settings and Information databases, to the Signaling and Message Transfer facilities.
- Multi-lingual Skins for customizable HTML, WAP/WML, and I-Mode Interfaces.
- XIMSS interface providing access to Mailbox, Groupware and File Stores, to Settings and Information

databases, to the Signaling and Message Transfer facilities.

- HTTP, WebDAV, FTP, and TFTP access to Account File Stores.
- AirSync interface to E-mail and Groupware data for Microsoft® Windows Mobile clients (server-based ActiveSync).
- CalDAV interface to Calendar and Tasks Mailboxes.
- Publish/Subscribe (`WebCal/iCal`) HTTP-based operations with Calendar and Tasks mailboxes.
- CardDAV interface to Contacts Mailboxes.
- ACAP access to the Account Information database.

## Advanced Security

- SASL Secure Authentication methods.
- GSSAPI (including Kerberos V5) authentication, SSO (single sign-on), and security.
- Client Certificate-based Secure Authentication methods.
- Secure Mail (S/MIME) WebMail implementation (encryption/decryption, digital signing, signature verification).
- Automatic Encryption implementing secure information storage.
- SSL/TLS Secure Transfer for SMTP, SIP, IMAP, POP, HTTP, LDAP, ACAP, PWD and Administration sessions.
- Lawful Interception functionality.

## Multi-tier Administration

- WebAdmin interface for administration, provisioning, and monitoring.
- CLI/API interface for administration, provisioning, and monitoring.
- SNMP Agent for remote monitoring.
- Triggers for proactive monitoring.
- Poppwd protocol for remote password modification.
- LDAP-based Provisioning (optional) for integration with legacy systems.
- BSD syslog Server to consolidate log records from third-party components.

## Multi-Server Operation

- Distributed Domains for Distributed multiple single-Server configurations.
- Static Clusters for Multi-Server Account partitioning.
- Dynamic Clusters for advanced scalability without Account partitioning. Carrier-grade 99.999%-uptime, field-proven for more than 5,000,000 real active Accounts.
- Cluster of Clusters for extra-large sites (over 10,000,000 active Accounts).

# Features

The CommuniGate Pro Server is based on the Internet Standards (RFCs), and it has many additional features required for today's industrial-level and carrier-grade communication systems. The Features table can be used to compare the CommuniGate Pro with other systems available on the market today.

# Administration

CommuniGate Pro Server can be configured and managed remotely (via the Internet) using any Web browser.

Remote administration features include:

- configuring the Server, Router, and all communication modules;
- creating, removing and updating of user account information;
- monitoring modules activity;
- monitoring System Logs;
- working with Server queues and individual messages in the Server queues.

# Support and Discussions

Please subscribe to the CGatePro@communigate.com mailing list to discuss the CommuniGate Pro related issues.
To subscribe to this mailing list, please send any message to CGatePro-on@communigate.com.
Since the traffic on this mailing list is rather high, you may want to subscribe in the Digest mode. To subscribe to the List in the Digest mode, send a message to CGatePro-digest@communigate.com
A searchable archive of the CommuniGate Pro list is available at
http://lists.communigate.com/Lists/CGatePro/List.html

If you do not feel comfortable sending your questions to the mailing list, we will promptly answer your letters sent to support@communigate.com.

CommuniGate Systems contact and general information is available at http://www.communigate.com.

# Updates and Bug Fixes

CommuniGate Pro is updated on a regular basis.

You can review the history of updates and bug fixes at the CommuniGate Pro History page.

# Download the Latest Versions

The CommuniGate Pro software is being updated on a regular basis.
The Stable versions include the latest Major Release and subsequent bug fix releases.
The Current versions contain new features that will appear in the next Major release.

Both Stable and Current versions can be used in the Trial Mode.
To use them in the production mode, your installation should have the License Keys valid for these versions.

| The Stable Release | | | |
|---|---|---|---|
| Operating System | CPU | Download | |
| | | via http | via ftp |
| Linux | x86 (rpm) | | |
| | x86-64 (rpm) | | |
| | x86 (deb) | | |
| | x86-64 (deb) | | |
| | PowerPC | | |
| | ARM (Raspberry Pi) | | |
| | Sparc | | |
| | MIPS | | |

| | | | |
|---|---|---|---|
| | MIPSEL | 🖫 | 🖫 |
| Linux (tgz archive) | x86 | 🖫 | 🖫 |
| Microsoft Windows | x86 | 🖫 | 🖫 |
| | x86-64 | 🖫 | 🖫 |
| Apple MacOS X | PowerPC | 🖫 | 🖫 |
| | x86 | 🖫 | 🖫 |
| FreeBSD 11.x | x86 | 🖫 | 🖫 |
| | x86-64 | 🖫 | 🖫 |
| FreeBSD 10.x | x86 | 🖫 | 🖫 |
| | x86-64 | 🖫 | 🖫 |
| FreeBSD 9.x | x86 | 🖫 | 🖫 |
| | x86-64 | 🖫 | 🖫 |
| Sun Solaris | Sparc | 🖫 | 🖫 |
| | x86 | 🖫 | 🖫 |
| IBM AIX | PowerPC | 🖫 | 🖫 |
| | PowerPC 64bit | 🖫 | 🖫 |
| **The Current ("cutting edge") Release** | | | |
| All Platforms | | LIST | |
| **The Retired Platforms** | | | |
| IBM OS/400 | AS/400 Power | 🖫 | 🖫 |
| QNX | x86 | 🖫 | 🖫 |
| OpenVMS | Alpha | 🖫 | 🖫 |
| | Itanium | 🖫 | 🖫 |
| SGI IRIX | MIPS | 🖫 | 🖫 |
| SCO UnixWare | x86 | 🖫 | 🖫 |
| SCO OpenServer | x86 | 🖫 | 🖫 |
| Tru64 | Alpha | 🖫 | 🖫 |
| IBM OS/2 | x86 | 🖫 | 🖫 |
| BSDI BSD/OS | x86 | 🖫 | 🖫 |
| BeOS | x86 | 🖫 | 🖫 |
| | PowerPC | 🖫 | 🖫 |
| Linux | IBM S/390 (31-bit) | 🖫 | 🖫 |
| | IBM S/390 (64-bit) | 🖫 | 🖫 |
| | Alpha | 🖫 | 🖫 |
| | Motorola 68K | 🖫 | 🖫 |
| | Itanium | 🖫 | 🖫 |
| HP/UX | HPPA | 🖫 | 🖫 |
| | Itanium | 🖫 | 🖫 |
| NetBSD | x86 | 🖫 | 🖫 |
| OpenBSD | x86 | 🖫 | 🖫 |
| Microsoft Windows | Itanium | 🖫 | 🖫 |

A copy of this Guide is included into all versions of the CommuniGate Pro software and can be accessed either via your CommuniGate Pro HTTP server, or directly, as HTML files in your Server WebGuide directory.

You can download the PDF version of this Guide.

# Download the CommuniGate Pro E-mail Plugins

CommuniGate Pro can employ many third-party E-mail Plugins. The following Plugins are supported by CommuniGate Systems:

- MailShell® SpamCatcher Plugin for CommuniGate Pro.
- Cloudmark® Anti-Spam Plugin for CommuniGate Pro.
- Kaspersky® Virus Scanner Plugin for CommuniGate Pro.
- McAfee® Virus Scanner Plugin for CommuniGate Pro.
- Sophos® Virus Scanner Plugin for CommuniGate Pro.

**Note:** These Plugins are available for certain platforms only.

# Download the CommuniGate Pro MAPI Connector

The CommuniGate Pro MAPI Connector can be installed on Microsoft® Windows workstations. It provides access to CommuniGate Pro Accounts via the Microsoft Messaging API (MAPI) - the method used with the Microsoft Outlook® and other MAPI-enabled applications.

| Version | Operating System | CPU | Download | |
|---|---|---|---|---|
| | | | via http | via ftp |
| Classic | Microsoft Windows | x86 | | |
| | Microsoft Windows | x86-64 | | |
| Sync (beta) | Microsoft Windows | x86 | | |
| | Microsoft Windows | x86-64 | | |

See the MAPI section for the installation and configuration instructions.

# Download the CommuniGate Media Plugins for Browsers

The CommuniGate Media Plugins are available for the popular modern Web browsers. The Plugins allow a Web-based application (such as Pronto! or an AJAX-based client) to implement a VoIP endpoint (a "softphone").

See the CommuniGate Media Plugin site for more details.

**CommuniGate Pro**

| Introduction | Features | History | How To | Help Me |
|---|---|---|---|---|

# Features and Standards

- **Kernel**
- **Security**
- **International**
- **Generic E-mail**
- **Mail Transfer**
  - **SMTP (Simple Mail Transfer Protocol)**
  - **LMTP (Local Mail Transfer Protocol)**
  - **Mailing Lists**
- **Signaling**
  - **SIP (Session Initiation Protocol)**
  - **XMPP (Extensible Messaging and Presence Protocol)**
  - **SMPP (Short Message Peer to Peer Protocol)**
  - **Simple Notification**
- **Data Access**
  - **IMAP (Internet Message Access Protocol)**
  - **POP and RPOP (Post Office Protocol)**
  - **FTP (File Transfer Protocol)**
  - **TFTP (Trivial File Transfer Protocol)**
  - **ACAP (Application Configuration Access Protocol)**
  - **WebUser Interface**
  - **WebDAV (WWW Distributed Authoring and Versioning)**
- **Multimedia**
  - **SDP (Session Description Protocol)**
  - **RTP (Transport Protocol for Real-Time Applications)**
  - **Voice and Video Mail**
- **Groupware**
  - **Calendar and Tasks**
  - **Contacts**
- **Services**
  - **HTTP (HyperText Transfer Protocol)**
  - **LDAP (Lightweight Directory Access Protocol)**
  - **SNMP (Simple Network Management Protocol)**
  - **RADIUS (Remote Authentication Dial In User Service)**
  - **STUN**
  - **BSD syslog**
  - **DNR (Domain Name Resolver)**
- **Universal**
  - **XIMSS (XML Interface to Messaging, Scheduling and Signaling)**

The CommuniGate Pro is based on Internet standards (RFCs). Additionally, it has many unique capabilities that have quickly become the "must-have" features for modern carrier-grade communication systems.

# Kernel

| Supported Standards | |
|---|---|
| RFC6713 | The 'application/zlib' and 'application/gzip' Media Types.<br>J. Levine. August 2012. |
| RFC6657 | Update to MIME regarding "charset" Parameter Handling<br>in Textual Media Types.<br>A. Melnikov, J. Reschke. July 2012. |
| RFC6598 | IANA-Reserved IPv4 Prefix for Shared Address Space.<br>J. Weil, V. Kuarsingh, C. Donley, C. Liljenstolpe, M. Azinger. April 2012. |
| RFC5735 | Special-Use IPv4 Addresses.<br>M. Cotton, L. Vegoda. January 2010. |
| RFC5137 | ASCII Escaping of Unicode Characters<br>J. Klensin. February 2008. |
| RFC4291 | IP Version 6 Addressing Architecture<br>R. Hinden, S. Deering. February 2006. |
| RFC4627 | The application/json Media Type for JavaScript Object Notation (JSON)<br>D. Crockford. July 2005. |
| RFC4021 | Registration of Mail and MIME Header Fields.<br>G. Klyne, J. Palme. March 2005. |
| RFC3986 | Uniform Resource Identifiers (URI): Generic Syntax.<br>T. Berners-Lee, R. Fielding, L. Masinter. January 2005. |
| RFC3548 | The Base16, Base32, and Base64 Data Encodings.<br>S. Josefsson, Ed. July 2003. |
| RFC3330 | Special-Use IPv4 Addresses.<br>IANA. September 2002. |
| RFC2732 | Format for Literal IPv6 Addresses in URL's.<br>R. Hinden, B. Carpenter, L. Masinter.December 1999. |
| RFC2392 | Content-ID and Message-ID Uniform Resource Locators.<br>E. Levinson. August 1998. |
| RFC2387 | The MIME Multipart/Related Content-type.<br>E. Levinson. August 1998. |
| RFC2231 | MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations.<br>N. Freed, K. Moore. November 1997. |
| RFC2047 | Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header Extensions for Non-ASCII Text.<br>K. Moore. November 1996. |
| RFC2046 | Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types.<br>N. Freed & N. Borenstein. November 1996. |
| RFC2045 | Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies.<br>N. Freed & N. Borenstein. November 1996. |
| RFC2044 | UTF-8, a transformation format of Unicode and ISO 10646<br>Yergeau, F. October 1996. |
| RFC1700/STD0002 | ASSIGNED NUMBERS.<br>J. Reynolds,J. Postel. October 1994. |
| RFC1123 | Requirements for Internet Hosts -- Application and Support<br>R. Braden, Editor. October 1989. |

# Security

| Supported Standards | |
|---|---|
| RFC6376 | DomainKeys Identified Mail (DKIM) Signatures<br>D. Crocker, T. Hansen, M. Kucherawy. September 2011. |
| RFC6347 | Datagram Transport Layer Security Version 1.2<br>E. Rescorla, N. Modadugu. January 2012. |
| RFC6302 | Logging Recommendations for Internet-Facing Servers<br>A. Durand, I. Gashinsky, D. Lee, S. Sheppard. June 2011. |
| RFC6234 | US Secure Hash Algorithms<br>(SHA and SHA-based HMAC and HKDF)<br>D. Eastlake 3rd, T. Hansen. May 2011. |
| RFC6176 | Prohibiting Secure Sockets Layer (SSL) Version 2.0.<br>S. Turner, T. Polk. March 2011. |
| RFC6101 | The Secure Sockets Layer (SSL) Protocol Version 3.0<br>A. Freier, P. Karlton, P. Kocher. August 2011. |
| RFC6066 | Transport Layer Security (TLS) Extensions: Extension Definitions<br>D. Eastlake 3rd. January 2011. |
| RFC5915 | Elliptic Curve Private Key Structure<br>S. Turner, D. Brown. June 2010. |
| RFC5746 | Transport Layer Security (TLS) Renegotiation Indication Extension.<br>E. Rescorla, M. Ray, S. Dispensa, N. Oskov. February 2010. |
| RFC5705 | Keying Material Exporters for Transport Layer Security (TLS).<br>E. Rescorla. March 2010. |
| RFC5480 | Elliptic Curve Cryptography Subject Public Key Information.<br>S. Turner, D. Brown, K. Yiu, R. Housley, T. Polk. March 2009. |
| RFC5289 | TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)<br>E. Rescorla. August 2008. |
| RFC5246 | The Transport Layer Security (TLS) Protocol. Version 1.2<br>T. Dierks, E. Rescorla. August 2008. |
| RFC4757 | The RC4-HMAC Kerberos Encryption Types Used by Microsoft Windows.<br>K. Jaganathan, L. Zhu, J. Brezak. December 2006. |
| RFC4752 | The Kerberos V5 ("GSSAPI") Simple Authentication and Security Layer (SASL) Mechanism.<br>A. Melnikov, Ed. November 2006. |
| RFC4616 | The PLAIN Simple Authentication and Security Layer (SASL) Mechanism.<br>K. Zeilenga, Ed. August 2006. |
| RFC4559 | SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows.<br>K. Jaganathan, L. Zhu, J. Brezak. June 2006. |
| RFC4492 | Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS).<br>S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, B. Moeller. May 2006 |
| RFC4422 | Simple Authentication and Security Layer (SASL).<br>A. Melnikov, K. Zeilenga, Ed. June 2006. |
| RFC4366 | Transport Layer Security (TLS) Extensions<br>M. Nystrom, D. Hopwood, J. Mikkelsen, T. Wright. April 2006. |
| RFC4347 | Datagram Transport Layer Security.<br>E. Rescorla, T. Dierks. April 2006. |

| | |
|---|---|
| RFC4346 | The Transport Layer Security (TLS) Protocol Version 1.1.<br>T. Dierks, E. Rescorla. April 2006. |
| RFC4178 | The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism.<br>Simple Authentication and Security Layer (SASL).<br>L. Zhu, P. Leach, K. Jaganathan, W. Ingersoll. October 2005. |
| RFC4121 | The Kerberos Version 5. Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2.<br>L. Zhu, K. Jaganathan, S. Hartman. July 2005. |
| RFC4120 | The Kerberos Network Authentication Service (V5).<br>C. Neuman, T. Yu, S. Hartman, K. Raeburn. July 2005. |
| RFC4055 | Tdditional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.<br>J. Schaad, B. Kaliski, R. Housley. June 2005. |
| RFC3961 | Encryption and Checksum Specifications for Kerberos 5.<br>K. Raeburn. February 2005. |
| RFC3546 | Transport Layer Security (TLS) Extensions.<br>S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, T. Wright. June 2003. |
| RFC3447 | Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1<br>J. Jonsson, B. Kaliski. February 2003. |
| RFC3280 | Internet X.509 Public Key Infrastructure. Certificate and CRL Profile.<br>W. Polk, W. Ford, D. Solo. April 2002. |
| RFC3279 | Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.<br>W. Polk, R. Housley, L. Bassham. April 2002. |
| RFC3268 | Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS).<br>P. Chown. June 2002. |
| RFC3174 | US Secure Hash Algorithm 1 (SHA1).<br>D. Eastlake, P. Jones. September 2001. |
| RFC2831 | Using Digest Authentication as a SASL Mechanism.<br>P. Leach, C. Newman. May 2000. |
| RFC2633 | S/MIME Version 3 Message Specification.<br>B. Ramsdell. June 1999. |
| RFC2632 | S/MIME Version 3 Certificate Handling.<br>B. Ramsdell. June 1999. |
| RFC2630 | Cryptographic Message Syntax.<br>R. Housley. June 1999. |
| RFC2617 | HTTP Authentication. Basic and Digest Access Authentication<br>J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart. June 1999. |
| RFC2595 | Using TLS with IMAP, POP3 and ACAP.<br>C. Newman. June 1999. |
| RFC2585 | Internet X.509 Public Key Infrastructure. Operational Protocols: FTP and HTTP<br>R. Housley, P. Hoffman. May 1999. |
| RFC2486 | The Network Access Identifier.<br>B. Aboba, M. Beadles. January 1999. |
| RFC2478 | The Simple and Protected GSS-API Negotiation Mechanism.<br>E. Baize, D. Pinkas. December 1998. |

| | |
|---|---|
| RFC2315 | PKCS #7: Cryptographic Message Syntax. Version 1.5<br>B. Kaliski. March 1998. |
| RFC2246 | The TLS Protocol. Version 1.0<br>T. Dierks. C. Allen, January 1999. |
| RFC2222 | Simple Authentication and Security Layer.<br>J. Myers. October 1997. |
| RFC2195 | IMAP/POP AUTHorize extension for Simple Challenge/Response.<br>J. Klensin & others. September 1997. |
| RFC2078 | Generic Security Service Application Program Interface, Version 2<br>J. Linn. January 1997. |
| RFC2104 | HMAC: Keyed-Hashing for Message Authentication.<br>H. Krawczyk, M.Bellare, R.Canetti. February 1997. |
| RFC1964 | The Kerberos Version 5 GSS-API Mechanism.<br>J. Linn. June 1996. |
| RFC1731 | IMAP4 Authentication Mechanisms.<br>J. Myers. December 1994. |
| RFC1510 | The Kerberos Network Authentication Service (V5)<br>J. Kohl, C. Neuman. September 1993. |
| RFC1321 | The MD5 Message-Digest Algorithm.<br>R. Rivest. April 1992. |

# International

| Supported Standards | |
|---|---|
| RFC3629 | UTF-8, a transformation format of ISO 10646.<br>F. Yergeau. November 2003. |
| RFC3492 | Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA).<br>A. Costello. March 2003. |
| RFC3490 | Internationalizing Domain Names in Applications (IDNA).<br>P. Faltstrom, P. Hoffman, A. Costello. March 2003. |
| RFC2319 | Ukrainian Character Set KOI8-U.<br>KOI8-U Working Group. April 1998. |
| RFC2278 | IANA Charset Registration Procedures.<br>N. Freed, J. Postel. January 1998. |
| RFC2184 | MIME Parameter Value and Encoded Word Extensions:<br>Character Sets, Languages, and Continuations.<br>N. Freed, K. Moore. August 1997. |
| RFC2152 | UTF-7<br>A Mail-Safe Transformation Format of Unicode.<br>D. Goldsmith. M. Davis. May 1997. |
| RFC1557 | Korean Character Encoding for Internet Messages.<br>U. Choi. K. Chon,H. Park. December 1993. |
| RFC1554 | ISO-2022-JP-2: Multilingual Extension of ISO-2022-JP.<br>M. Ohta, K. Handa. December 1993. |
| RFC1489 | Registration of a Cyrillic Character Set.<br>A. Chernov. July 1993. |
| RFC1468 | Japanese Character Encoding for Internet Messages. |

J. Murai, M. Crispin, E. van der Poel. June 1993.

# Generic E-mail

| Supported Standards | |
|---|---|
| RFC1892/RFC3462/RFC6522 | The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages.<br>M. Kucherawy, Ed. January 2012. |
| RFC5598 | Internet Mail Architecture.<br>D. Crocker. July 2009. |
| RFC5423 | Internet Message Store Events.<br>R. Gellens, C. Newman. March 2009. |
| RFC3676 | The Text/Plain Format and DelSp Parameters.<br>R. Gellens. February 2004. |
| RFC2183 | Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field.<br>R. Troost, S. Dorner, K. Moore. August 1997. |
| RFC2111 | Content-ID and Message-ID Uniform Resource Locators.<br>E. Levinson. March 1997. |
| RFC2076 | Common Internet Message Headers.<br>J. Palme. February 1997. |
| RFC1894/RFC3464 | An Extensible Message Format for Delivery Status Notifications.<br>K. Moore & G. Vaudreuil. January 1996. |
| RFC1740 | MIME Encapsulation of Macintosh files - MacMIME.<br>P. Faltstrom, D. Crocker, E. Fair. December 1994. |
| RFC1741 | MIME Content Type for BinHex Encoded Files.<br>P. Faltstrom, D. Crocker, E. Fair. December 1994. |
| RFC1711 | Classifications in E-mail Routing.<br>J. Houttuin. October 1994. |
| RFC0822/STD0011/RFC2822/RFC5322 | Standard for the format of ARPA Internet text messages/Internet Message Format.<br>D.Crocker/P. Resnick, Ed. Aug-13-1982/April 2001/October 2008. |

# Mail Transfer

## SMTP

| Supported Standards | |
|---|---|
| RFC7489 | Domain-based Message Authentication, Reporting, and Conformance (DMARC)<br>M. Kucherawy, E. Zwicky. March 2015. |
| RFC6531 | SMTP Extension for Internationalized Email<br>J. Yao, W. Mao. February 2012. |
| RFC4408 | Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1.<br>M. Wong, W. Schlitt. April 2006. |
| RFC3865 | A No Soliciting Simple Mail Transfer Protocol (SMTP) |

| | Service Extension. C. Malamud. September 2004. |
|---|---|
| RFC3848 | ESMTP and LMTP Transmission Types Registration. C. Newman. July 2004. |
| RFC3461 | SMTP Service Extension for Delivery Status Notifications. K. Moore. January 2003. |
| RFC3207 | SMTP Service Extension for Secure SMTP over Transport Layer Security (TLS). P. Hoffman. February 2002. |
| RFC2920/ RFC2197 | SMTP Service Extension for Command Pipelining. N. Freed. September 2000. |
| RFC2645 | ON-DEMAND MAIL RELAY (ODMR) SMTP with Dynamic IP Addresses R. Gellens. August 1999. |
| RFC2554 | SMTP Service Extension for Authentication. J. Myers. March 1999. |
| RFC2505 | Anti-Spam Recommendations for SMTP MTAs. G. Lindberg. February 1999. |
| RFC2476 | Message Submission. R. Gellens, J. Klensin. December 1998. |
| RFC1985 | SMTP Service Extension for Remote Message Queue Starting. J. De Winter. August 1996. |
| RFC1870 | SMTP Service Extension for Message Size Declaration. J. Klensin, N. Freed, & K. Moore. November 1995. |
| RFC1869 | SMTP Service Extensions. J. Klensin, N. Freed, M. Rose, E. Stefferud & D. Crocker. November 1995. |
| RFC1652 | SMTP Service Extension for 8bit-MIMEtransport. J. Klensin, N. Freed, M. Rose, E. Stefferud & D. Crocker. July 1994. |
| RFC0974 | Mail routing and the domain system. C. Partridge. Jan-01-1986. |
| RFC0821/STD0010/RFC2821/RFC5321 | Simple Mail Transfer Protocol. J. Postel. Aug-01-1982/April 2001/J. Klensin. October 2008 |

## LMTP

| Supported Standards | |
|---|---|
| RFC3848 | ESMTP and LMTP Transmission Types Registration. C. Newman. July 2004. |
| RFC2033 | Local Mail Transfer Protocol. J. Myers. October 1996. |

## Mailing Lists

| Supported Standards | |
|---|---|
| RFC2919 | List-Id: A Structured Field and Namespace for the Identification of Mailing Lists. R. Chandhok, G. Wenger. March 2001 |
| RFC2369 | The Use of URLs as Meta-Syntax for Core Mail List Commands |

| | and their Transport through Message Header Fields<br>G. Neufeld, J. Baer. July 1998 |
|---|---|
| [RFC1153](#) | Digest Message Format.<br>F. Wancho. April 1990. |

# Signaling

## SIP

| Supported Standards | |
|---|---|
| [RFC6337](#) | SIP Usage of the Offer/Answer Model.<br>S. Okumura, T. Sawada, P. Kyzivat. August 2011. |
| [RFC5627](#) | Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP).<br>J. Rosenberg. October 2009. |
| [RFC4480](#) | RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF).<br>H. Schulzrinne, V. Gurbani, P. Kyzivat, J. Rosenberg. July 2006. |
| [RFC4244](#) | An Extension to the Session Initiation Protocol (SIP) for Request History Information<br>M. Barnes, Ed. November 2005. |
| [RFC4235](#) | An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP)<br>J. Rosenberg, H. Schulzrinne, R. Mahy, Ed. November 2005. |
| [RFC4028](#) | Session Timers in the Session Initiation Protocol (SIP)<br>S. Donovan, J. Rosenberg. April 2005. |
| [RFC3966](#) | The tel URI for Telephone Numbers<br>H. Schulzrinne. December 2004. |
| [RFC3960](#) | Early Media and Ringing Tone Generation in the Session Initiation Protocol (SIP)<br>G. Camarillo, H. Schulzrinne. December 2004. |
| [RFC3903](#) | The Session Initiation Protocol (SIP) Extension for Event State Publication.<br>A. Niemi, Ed. October 2004. |
| [RFC3892](#) | The Session Initiation Protocol (SIP) Referred-By Mechanism.<br>R. Sparks. September 2004. |
| [RFC3891](#) | The Session Initiation Protocol (SIP) "Replaces" Header.<br>R. Mahy, B. Biggs, R. Dean. September 2004. |
| [RFC3863](#) | Presence Information Data Format (PIDF).<br>H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, J. Peterson. August 2004. |
| [RFC3858](#) | An Extensible Markup Language (XML) Based Format for Watcher Information.<br>J. Rosenberg. August 2004. |
| [RFC3857](#) | A Watcher Information Event Template-Package for<br>the Session Initiation Protocol (SIP).<br>J. Rosenberg. August 2004. |
| [RFC3856](#) | A Presence Event Package for the Session Initiation Protocol (SIP).<br>J. Rosenberg. August 2004. |
| [RFC3842](#) | A Message Summary and Message Waiting Indication Event Package for<br>the Session Initiation Protocol (SIP).<br>R. Mahy. August 2004. |
| [RFC3824](#) | Using E.164 numbers with the Session Initiation Protocol (SIP)<br>J. Peterson, H. Liu, J. Yu, B. Campbell. June 2004. |
| [RFC3690](#) | IP Telephony Requirements for Emergency Telecommunication Service (ETS) |

| | K. Carlberg, R. Atkinson. February 2004. |
|---|---|
| RFC3689 | General Requirements for Emergency Telecommunication Service (ETS)<br>K. Carlberg, R. Atkinson. February 2004. |
| RFC3581 | An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing.<br>J. Rosenberg, H. Schulzrinne. August 2003. |
| RFC3515 | The SIP Refer Method.<br>R. Sparks. April 2003. |
| RFC3428 | Session Initiation Protocol (SIP) Extension for Instant Messaging.<br>B. Campbell, Ed., J. Rosenberg, H. Schulzrinne, C. Huitema, D. Gurle. December 2002. |
| RFC3420 | Internet Media Type message/sipfrag.<br>R. Sparks. November 2002. |
| RFC3372 | Session Initiation Protocol for Telephones (SIP-T): Context and Architectures.<br>A. Vemuri, J. Peterson. September 2002. |
| RFC3327 | Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts.<br>D. Willis, B. Hoeneisen. December 2002. |
| RFC3326 | The Reason Header Field for the Session Initiation Protocol (SIP).<br>H. Schulzrinne, D. Oran, G. Camarillo. December 2002. |
| RFC3325 | Private Extensions to the Session Initiation Protocol (SIP) for<br>Asserted Identity within Trusted Networks.<br>C. Jennings, J. Peterson, M. Watson. November 2002. |
| RFC3323 | A Privacy Mechanism for the Session Initiation Protocol (SIP).<br>J. Peterson. November 2002. |
| RFC3311 | The Session Initiation Protocol (SIP) UPDATE Method.<br>J. Rosenberg. September 2002. |
| RFC3265 | Session Initiation Protocol (SIP)-Specific Event Notification.<br>B. Roach. June 2002. |
| RFC3263 | Session Initiation Protocol (SIP): Locating SIP Servers.<br>J. Rosenberg, H. Schulzrinne. June 2002. |
| RFC3262 | Reliability of Provisional Responses in Session Initiation Protocol (SIP).<br>J. Rosenberg, H. Schulzrinne. June 2002. |
| RFC3261 | SIP: Session Initiation Protocol.<br>J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler. June 2002. |
| RFC2976 | The SIP INFO Method.<br>S. Donovan. October 2000. |
| [MICROSOFT] | Microsoft SIP Message signing |

## XMPP

| Supported Standards | |
|---|---|
| RFC3921 | Extensible Messaging and Presence Protocol (XMPP):<br>Instant Messaging and Presence.<br>P. Saint-Andre, Ed. October 2004. |
| RFC3920 | Extensible Messaging and Presence Protocol (XMPP): Core<br>P. Saint-Andre, Ed. October 2004. |
| XEP0243 | XMPP Server Compliance 2009.<br>Peter Saint-Andre. |
| XEP0203 | Delayed Delivery. |

| | | |
|---|---|---|
| | Peter Saint-Andre. | |
| XEP0202 | Entity Time. Peter Saint-Andre, Maciej Niedzielski. | |
| XEP0199 | XMPP Ping. Peter Saint-Andre. | |
| XEP0160 | Best Practices for Handling Offline Messages. Peter Saint-Andre. | |
| XEP0153 | vCard-Based Avatars. Peter Saint-Andre. | |
| XEP0114 | Jabber Component Protocol. Peter Saint-Andre. | |
| XEP0092 | Software Version. Peter Saint-Andre. | |
| XEP0090 | Entity Time. Peter Saint-Andre. | |
| XEP0086 | Error Condition Mappings. Robert Norris, Peter Saint-Andre. | |
| XEP0085 | Chat State Notifications. Peter Saint-Andre, Dave Smith. | |
| XEP0082 | XMPP Date and Time Profiles. Peter Saint-Andre. | |
| XEP0078 | Non-SASL Authentication. Peter Saint-Andre. | |
| XEP0077 | In-Band Registration. Peter Saint-Andre. | |
| XEP0076 | Malicious Stanzas. Peter Saint-Andre, Joe Hildebrand. | |
| XEP0055 | Jabber Search. Peter Saint-Andre. | |
| XEP0054 | vcard-temp. Peter Saint-Andre. | |
| XEP0049 | Private XML Storage. Peter Saint-Andre, Russell Davis. | |
| XEP0045 | Multi-User Chat. Peter Saint-Andre. | |
| XEP0030 | Service Discovery. Joe Hildebrand, Peter Millard, Ryan Eatmon, Peter Saint-Andre. | |
| XEP0022 | Message Events. Jeremie Miller, DJ Adams, Peter Saint-Andre. | |
| XEP0012 | Last Activity. Jeremie Miller, Thomas Muldowney, Peter Saint-Andre. | |
| XEP0011 | Jabber Browsing. Jeremie Miller, Julian Missig, Thomas Muldowney. | |
| XEP0008 | IQ-Based Avatars. Thomas Muldowney, Julian Missig, Jens Alfke, Peter Millard. | |

## SMPP

| Supported Standards |
|---|
| |

| Supported Standards | |
|---|---|
| SMPP 3.4 | Short Message Peer to Peer Protocol Specification v3.4. SMPP Developers Forum. October 1999. |

## Simple Notification

| Supported Standards | |
|---|---|
| RFC4146 | Simple New Mail Notification. R. Gellens. August 2005. |

# Data Access

## IMAP

| Supported Standards | |
|---|---|
| RFC6851 | Internet Message Access Protocol (IMAP) - MOVE Extension. A. Gulbrandsen, N. Freed. January 2013 |
| RFC6154 | IMAP LIST Extension for Special-Use Mailboxes. B. Leiba, J. Nicolson. March 2011 |
| RFC5258 | Internet Message Access Protocol version 4 - LIST Command Extensions. B. Leiba, A. Melnikov. June 2008 |
| RFC4959 | IMAP Extension for Simple Authentication and Security Layer (SASL) Initial Client Response. R. Siemborski, A. Gulbrandsen. September 2007 |
| RFC4731 | IMAP4 Extension to SEARCH Command for Controlling What Kind of Information Is Returned. A. Melnikov, D. Cridland. November 2006 |
| RFC4466 | Collected Extensions to IMAP4 ABNF. A. Melnikov, C. Daboo. April 2006 |
| RFC4315 | Internet Message Access Protocol (IMAP) - UIDPLUS extension. M. Crispin, December 2005 |
| RFC3691 | Internet Message Access Protocol (IMAP) UNSELECT command. A. Melnikov, February 2004 |
| RFC3516 | IMAP4 Binary Content Extension. L. Nerenberg, April 2003 |
| RFC3503 | MDN profile for IMAP. A. Melnikov, March 2003 |
| RFC3502 | The Internet Message Action Protocol (IMAP4) MULTIAPPEND Extension. M. Crispin, March 2003 |
| RFC3348 | The Internet Message Action Protocol (IMAP4) Child Mailbox Extension. M. Gahrns, R. Cheng, July 2002 |
| RFC2971 | IMAP4 ID extension. T. Showalter, October 2000 |
| RFC2683 | IMAP4 Implementation Recommendations. B. Leiba, September 1999 |
| RFC2595 | Using TLS with IMAP, POP3 and ACAP. C. Newman. June 1999. |
| RFC2359 | IMAP4 UIDPLUS extension. |

| | |
|---|---|
| | J. Myers, June 1998 |
| RFC2342 | IMAP4 Namespace.<br>M. Gahrns, C. Newman, May 1998 |
| RFC2221 | IMAP4 Login Referrals.<br>M. Gahrns, October 1997 |
| RFC2192 | IMAP URL Scheme.<br>C. Newman. September 1997. |
| RFC2180 | IMAP4 Multi-Accessed Mailbox Practice.<br>M. Gahrns. July 1997. |
| RFC2177 | IMAP4 IDLE command.<br>B. Leiba. June 1997. |
| RFC2088 | IMAP4 non-synchronizing literals.<br>J. Myers. January 1997. |
| RFC2087 | IMAP4 QUOTA extension.<br>J. Myers, January 1997. |
| RFC2086 | IMAP4 ACL extension.<br>J. Myers, January 1997 |
| RFC2060/RFC3501 | INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1.<br>M. Crispin. December 1996/March 2003. |
| RFC1731 | IMAP4 Authentication Mechanisms.<br>J. Myers. December 1994. |
| RFC1730 | INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4.<br>M. Crispin. December 1994. |

## POP

| Supported Standards | |
|---|---|
| RFC5034 | The Post Office Protocol (POP3) Simple Authentication and Security Layer (SASL) Authentication Mechanism.<br>R. Siemborski, A. Menon-Sen. July 2007. |
| RFC2595 | Using TLS with IMAP, POP3 and ACAP.<br>C. Newman. June 1999. |
| RFC2449 | POP3 Extension Mechanism<br>R. Gellens, C. Newman, L. Lundblade. November 1998. |
| RFC2384 | POP URL Scheme<br>R. Gellens. August 1998. |
| RFC1939/RFC1725/STD0053 | Post Office Protocol - Version 3.<br>J. Myers & M. Rose. May 1996. |
| RFC1734 | POP3 AUTHentication command.<br>J. Myers. December 1994. |
| draft-hansen-pop3-xtndext-00.txt | POP3 XTND Extensions |
| | POP3 LAST command |

## FTP

| Supported Standards | |
|---|---|
| RFC5797 | FTP Command and Extension Registry |

| | J. Klensin, A. Hoenes. March 2010. |
|---|---|
| RFC4217 | Securing FTP with TLS<br>P. Ford-Hutchinson. October 2005. |
| RFC2428 | FTP Extensions for IPv6 and NATs<br>M. Allman, S. Ostermann, C. Metz. September 1998. |
| RFC2389 | Feature negotiation mechanism for the File Transfer Protocol<br>P. Hethmon, R. Elz. August 1998. |
| RFC2228 | FTP Security Extensions<br>M. Horowitz, S. Lunt. October 1997. |
| RFC0959 | FILE TRANSFER PROTOCOL (FTP)<br>J. Postel, J. Reynolds. October 1985. |
| RFC0775 | DIRECTORY ORIENTED FTP COMMANDS<br>David Mankins, Dan Franklin,A. D. Owen. June 1980. |

## TFTP

| Supported Standards | |
|---|---|
| RFC2349 | TFTP Timeout Interval and Transfer Size Options<br>G. Malkin, A. Harkin. May 1998. |
| RFC2348 | TFTP Blocksize Option<br>G. Malkin, A. Harkin. May 1998. |
| RFC2347 | TFTP Option Extension<br>G. Malkin, A. Harkin. May 1998. |
| RFC1350 | THE TFTP PROTOCOL (REVISION 2)<br>K. Sollins. July 1992. |

## ACAP

| Supported Standards | |
|---|---|
| RFC2595 | Using TLS with IMAP, POP3 and ACAP.<br>C. Newman. June 1999. |
| RFC2244 | Application Configuration Access Protocol.<br>C. Newman, J. Myers. November 1997. |

## WebUser Interface

| Supported Standards | |
|---|---|
| RFC6068 | The 'mailto' URI Scheme.<br>M. Duerst, L. Masinter, J. Zawinski. October 2010. |
| RFC2557 | MIME Encapsulation of Aggregate Documents, such as HTML (MHTML).<br>J. Palme, A. Hopmann, N. Shelness. March 1999. |
| RFC2368 | The mailto URL scheme.<br>P. Hoffman, L. Masinter, J. Zawinski. July 1998. |
| RFC2298 | An Extensible Message Format for Message Disposition Notifications.<br>R. Fajman. March 1998. |
| RFC1896 | The text/enriched MIME Content-type.<br>P. Resnick,A. Walker. February 1996. |
| | |

| | |
|---|---|
| RFC1844 | Multimedia E-mail (MIME) User Agent checklist.<br>E. Huizer. August 1995. |
| RFC1808 | Relative Uniform Resource Locators.<br>R. Fielding. June 1995. |
| RFC1738 | Uniform Resource Locators (URL).<br>T. Berners-Lee, L. Masinter, M. McCahill. December 1994. |

## WebDAV

| Supported Standards | |
|---|---|
| RFC5689 | Extended MKCOL for Web Distributed Authoring and Versioning (WebDAV).<br>C. Daboo. September 2009. |
| RFC5397 | WebDAV Current Principal Extension.<br>W. Sanchez, C. Daboo. November 2008. |
| RFC5323 | Web Distributed Authoring and Versioning (WebDAV) SEARCH.<br>J. Reschke, S. Reddy, J. Davis, A. Babich. November 2008. |
| RFC4918 | HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV).<br>L. Dusseault. June 2007. |
| RFC4709 | Mounting Web Distributed Authoring and Versioning (WebDAV) Servers.<br>J. Reschke. October 2006. |
| RFC4331 | Quota and Size Properties for Distributed Authoring and Versioning (DAV) Collections.<br>B. Korver, L. Dusseault. February 2006. |
| RFC4316 | Datatypes for Web Distributed Authoring and Versioning (WebDAV) Properties.<br>J. Reschke. December 2005. |
| RFC3744 | Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol.<br>G. Clemm, J. Reschke, E. Sedlar, J. Whitehead. May 2004. |
| RFC3253 | Versioning Extensions to WebDAV(Web Distributed Authoring and Versioning)<br>J. Amsden, T. Ellison, C. Kaler, J. Whitehead. March 2002. |
| RFC2291 | Requirements for a Distributed Authoring and Versioning Protocol for the World Wide Web.<br>J. Slein, F. Vitali, E. Whitehead, D. Durand. February 1998. |

# MultiMedia

## SDP

| Supported Standards | |
|---|---|
| RFC4572 | Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)<br>J. Lennox. July 2006. |
| RFC4568 | Session Description Protocol (SDP). Security Descriptions for Media Streams<br>F. Andreasen, M. Baugher, D. Wing. July 2006. |
| RFC4566 | SDP: Session Description Protocol.<br>M. Handley, V. Jacobson, C. Perkins. July 2006. |
| RFC3605 | Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP).<br>C. Huitema. October 2003. |
| RFC3266 | Support for IPv6 in Session Description Protocol (SDP). |

| | S. Olson, G. Camarillo, A. B. Roach. June 2002. |
|---|---|
| RFC3264 | An Offer/Answer Model with Session Description Protocol (SDP).<br>J. Rosenberg, H. Schulzrinne. June 2002. |
| RFC2327 | SDP: Session Description Protocol.<br>M. Handley, V. Jacobson. April 1998. |

## RTP

| Supported Standards | |
|---|---|
| RFC5764 | Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP).<br>D. McGrew, E. Rescorla. May 2010. |
| RFC5763 | Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS).<br>J. Fischl, H. Tschofenig, E. Rescorla. May 2010. |
| RFC5761 | Multiplexing RTP Data and Control Packets on a Single Port.<br>C. Perkins, M. Westerlund. April 2010. |
| RFC5245 | Interactive Connectivity Establishment (ICE):<br>A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols<br>J. Rosenberg. April 2010. |
| RFC4961 | Symmetric RTP / RTP Control Protocol (RTCP).<br>D. Wing. July 2007. |
| RFC3711 | The Secure Real-time Transport Protocol (SRTP).<br>M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman. March 2004. |
| RFC3555 | MIME Type Registration of RTP Payload Formats.<br>S. Casner, P. Hoschka. July 2003. |
| RFC3551 | RTP Profile for Audio and Video Conferences with Minimal Control.<br>H. Schulzrinne, S. Casner. July 2003. |
| RFC3550 | RTP: A Transport Protocol for Real-Time Applications.<br>H. Schulzrinne, S. Casner, R. Frederick,V. Jacobson. July 2003. |
| RFC2833 | RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals.<br>H. Schulzrinne, S. Petrack. May 2000. |

## Voice and Video Mail

| Supported Standards | |
|---|---|
| RFC4024 | Voice Messaging Client Behaviour.<br>G. Parsons,J. Maruszak. July 2005. |
| RFC3938 | Video-Message Message-Context.<br>T. Hansen. October 2004. |
| RFC3773 | High-Level Requirements for Internet Voice Mail.<br>E. Candell. June 2004. |
| RFC3458 | Message Context for Internet Mail.<br>E. Burger, E. Candell, C. Eliot, G. Klyne. January 2003. |
| RFC2423 | VPIM Voice Message MIME Sub-type Registration.<br>G. Vaudreuil, G. Parsons. September 1998. |
| RFC2421 | Voice Profile for Internet Mail - version 2.<br>G. Vaudreuil, G. Parsons. September 1998. |

# Groupware

## Calendar and Tasks

| Supported Standards | |
|---|---|
| draft-desruisseaux-caldav-sched-08 | CalDAV Scheduling Extensions to WebDAV.<br>C. Daboo, B. Desruisseaux. August 2009. |
| draft-daboo-calendar-availability-01 | Calendar Availability.<br>C. Daboo, B. Desruisseaux. November 2008. |
| draft-caldav-ctag-02 | Calendar Collection Entity Tag (CTag) in CalDAV.<br>C. Daboo. May 2007. |
| RFC5546 | iCalendar Transport-Independent Interoperability Protocol (iTIP).<br>C. Daboo, Ed. December 2009. |
| RFC5545 | Internet Calendaring and Scheduling Core Object Specification (iCalendar).<br>B. Desruisseaux, Ed. September 2009. |
| RFC4791 | Calendaring Extensions to WebDAV (CalDAV).<br>C. Daboo, B. Desruisseaux, L. Dusseault. March 2007. |
| RFC3283 | Guide to Internet Calendaring.<br>B. Mahoney, G. Babics, A. Taler. June 2002. |
| RFC2447 | iCalendar Message-Based Interoperability Protocol (iMIP).<br>F. Dawson, S. Mansour, S. Silverberg. November 1998. |
| RFC2446 | iCalendar Transport-Independent Interoperability Protocol (iTIP).<br>S. Silverberg, S. Mansour, F. Dawson, R. Hopson. November 1998. |
| RFC2445 | Internet Calendaring and Scheduling Core Object Specification (iCalendar).<br>F. Dawson, D. Stenerson. November 1998. |

## Contacts

| Supported Standards | |
|---|---|
| RFC6352 | CardDAV: vCard Extensions to Web Distributed Authoring and Versioning (WebDAV).<br>C. Daboo. August 2011. |
| RFC6350 | vCard Format Specification.<br>S. Perreault. August 2011. |
| RFC4770 | vCard Extensions for Instant Messaging (IM).<br>C. Jennings, J. Reschke. January 2007. |
| RFC2426 | vCard MIME Directory Profile.<br>F. Dawson, T. Howes. September 1998. |
| RFC2425 | A MIME Content-Type for Directory Information.<br>T. Howes, M. Smith, F. Dawson. September 1998. |
| [VCARD] | vCard 2.1 |

# Services

## HTTP

| Supported Standards | |
|---|---|
| [RFC6585](#) | Additional HTTP Status Codes<br>M. Nottingham, R. Fielding. April 2012. |
| [RFC5785](#) | Defining Well-Known Uniform Resource Identifiers (URIs)<br>M. Nottingham, E. Hammer-Lahav. April 2010. |
| [RFC2818](#) | HTTP Over TLS<br>E. Rescorla. May 2000. |
| [RFC2817](#) | Upgrading to TLS Within HTTP/1.1<br>R. Khare, S. Lawrence. May 2000. |
| [RFC2617](#) | HTTP Authentication: Basic and Digest Access Authentication<br>J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart. June 1999. |
| [RFC2616](#) | Hypertext Transfer Protocol -- HTTP/1.1.<br>R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee. January 1997. |
| [RFC2388](#) | Returning Values from Forms: multipart/form-data<br>L. Masinter. August 1998. |
| [RFC2145](#) | Use and Interpretation of HTTP Version Numbers<br>J. C. Mogul, R. Fielding, J. Gettys, H. Frystyk. May 1997. |
| [RFC2109](#) | HTTP State Management Mechanism<br>D. Kristol, L. Montulli. February 1997. |

## LDAP

| Supported Standards | |
|---|---|
| [RFC2891](#) | LDAP Control Extension for Server Side Sorting of Search Results<br>T. Howes, M. Wahl, A. Anantha. August 2000. |
| [RFC2849](#) | The LDAP Data Interchange Format (LDIF) - Technical Specification<br>G. Good. June 2000. |
| [RFC2830](#) | LDAPv3: Extension for Transport Layer Security<br>J. Hodges, D. Byrne, B. Blakley, P. Behera. May 2000. |
| [RFC2829](#) | Authentication Methods for LDAP<br>M. Wahl, H. Alvestrand, J. Hodges, R. Morgan. May 2000. |
| [RFC2820](#) | Access Control Requirements for LDAP<br>E. Stokes, R. Morgan, M. Wahl. May 2000. |
| [RFC2798](#) | Definition of the inetOrgPerson LDAP Object Class.<br>M. Smith. April 2000. |
| [RFC2696](#) | LDAP Control Extension for Simple Paged Results Manipulation<br>A. Herron, A. Anantha, T. Howes. September 1999. |
| [RFC2587](#) | Internet X.509 Public Key Infrastructure.<br>LDAPv2 Schema<br>S. Boeyen, T. Howes, P. Richard. June 1999. |
| [RFC2256](#) | A Summary of the X.500(96) User Schema for use with LDAPv3.<br>M. Wahl. December 1997. |
| [RFC2255](#) | The LDAP URL Format.<br>T.Howes, M.Smith. December 1997. |
| [RFC2254](#) | The String Representation of LDAP Search Filters.<br>T. Howes. December 1997. |
| [RFC2253](#) | Lightweight Directory Access Protocol (v3): |

| | UTF-8 String Representation of Distinguished Names. M. Wahl, S. Kille, T. Howes. December 1997. |
|---|---|
| RFC2252 | Lightweight Directory Access Protocol (v3). Attribute Syntax Declarations. M. Wahl, T.Howes, S.Kille. December 1997. |
| RFC2251 | Lightweight Directory Access Protocol (v3). M. Wahl, T.Howes, S.Kille. December 1997. |
| RFC2247 | Using Domains in LDAP/X.500 Distinguished Names. S. Kille, M. Wahl, A. Grimstad, R. Huber, S. Sataluri. January 1998. |

## SNMP

| Supported Standards | |
|---|---|
| RFC2578 | Structure of Management Information Version 2 (SMIv2). K. McCloghrie, D. Perkins, J. Schoenwaelder. April 1999. |
| RFC1907 | Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2). SNMPv2 Working Group & others. January 1996. |
| RFC1906 | Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2). SNMPv2 Working Group & others. January 1996. |
| RFC1905 | Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2). SNMPv2 Working Group & others. January 1996. |
| RFC1904 | Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2). SNMPv2 Working Group & others. January 1996. |
| RFC1903 | Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2). SNMPv2 Working Group & others. January 1996. |
| RFC1902 | Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2). SNMPv2 Working Group & others. January 1996. |
| RFC1901 | Introduction to Community-based SNMPv2. SNMPv2 Working Group & others. January 1996. |
| RFC1212 | Concise MIB Definitions. Rose, M., and K. McCloghrie. March 1991. |
| RFC1157 | A Simple Network Management Protocol (SNMP). J. Case, M. Fedor, M. Schoffstall, J. Davin. May 1990. |

## RADIUS

| Supported Standards | |
|---|---|
| RFC4590 | RADIUS Extension for Digest Authentication B. Sterman, D. Sadolevsky, D. Schwartz, D. Williams, W. Beck. July 2006. |
| RFC3748 | PPP Extensible Authentication Protocol (EAP) B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz. June 2004. |
| RFC3579 | RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP) |

| | B. Aboba, P. Calhoun. September 2003. |
|---|---|
| RFC3079 | Deriving Keys for use with Microsoft Point-to-Point Encryption (MPPE)<br>G. Zorn. March 2001. |
| RFC2869 | RADIUS Extensions<br>C. Rigney, W. Willats, P. Calhoun. June 2000. |
| RFC2868 | RADIUS Attributes for Tunnel Protocol Support<br>G. Zorn, D. Leifer, A. Rubens, J. Shriver, M. Holdrege, I. Goyret. June 2000. |
| RFC2866 | RADIUS Accounting<br>C. Rigney. June 2000. |
| RFC2865 | Remote Authentication Dial In User Service (RADIUS)<br>C. Rigney, S. Willens, A. Rubens, W. Simpson. June 2000. |
| RFC2759 | Microsoft PPP CHAP Extensions, Version 2<br>G. Zorn. January 2000. |
| RFC2548 | Microsoft Vendor-specific RADIUS Attributes.<br>G. Zorn. March 1999. |
| RFC1994 | PPP Challenge Handshake Authentication Protocol (CHAP)<br>W. Simpson. August 1996. |

## STUN

| Supported Standards | |
|---|---|
| RFC5389 | Session Traversal Utilities for NAT (STUN).<br>J. Rosenberg, R. Mahy, P. Matthews, D. Wing. October 2008. |
| RFC3489 | STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs).<br>J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy. March 2003. |

## BSD syslog

| Supported Standards | |
|---|---|
| RFC6012 | The Syslog Protocol.<br>R. Gerhards. March 2009. |
| RFC3164 | The BSD syslog Protocol.<br>C. Lonvick. August 2001. |

## DNR

| Supported Standards | |
|---|---|
| RFC6116 | The E.164 to Uniform Resource Identifiers (URI)<br>.Dynamic Delegation Discovery System (DDDS) Application (ENUM)<br>S. Bradner, L. Conroy, K. Fujiwara. March 2011. |
| RFC5966 | DNS Transport over TCP - Implementation Requirements<br>R. Bellis. August 2010. |
| RFC3761 | The E.164 to Uniform Resource Identifiers (URI)<br>.Dynamic Delegation Discovery System (DDDS) Application (ENUM)<br>P. Faltstrom, M. Mealling. April 2004. |
| RFC3596 | DNS Extensions to Support IP Version 6<br>S. Thomson, C. Huitema, V. Ksinant, M. Souissi. October 2003. |

| | |
|---|---|
| RFC2916 | E.164 number and DNS.<br>P. Faltstrom. September 2000. |
| RFC2915 | The Naming Authority Pointer (NAPTR) DNS Resource Record.<br>M. Mealling, R. Daniel. September 2000. |
| RFC2782 | A DNS RR for specifying the location of services (DNS SRV).<br>A. Gulbrandsen, P. Vixie, L. Esibov. February 2000. |
| RFC1035 | Domain names - implementation and specification.<br>P.V.Mockapetris. Nov-01-1987. |

# Universal

## XIMSS

| Supported Standards | |
|---|---|
| XIMSS | XML Interface to Messaging, Scheduling and Signaling (XIMSS) protocol.<br>work-in-progress. |

CommuniGate **Pro**

| Introduction | Features | **History** | How To | Help Me |

# Version 6.2 Revision History

- **6.2: Summary**
- **6.1**
- **6.0**
- **5.4**
- **5.3**
- **5.2**
- **5.1**
- **5.0**
- **4.x**
- **3.x**
- **2.x**
- **1.x**

- **Pronto!**
- **MAPI Connector**

### Navigation (sidebar)

- Introduction
- Installation
- SysAdmin
- Network
- Objects
- Storage
- E-mail
- Real-Time
- Access
- Services
- Directory
- Clusters
- Applications
- Miscellaneous
- Licensing
- WebMail
- Pronto!
- PBX

RSS

Cluster upgrade note: for rolling upgrades, your Cluster must be upgraded to the 6.1.14 or better version before upgrading to any 6.2 version.

**6.2.2** 02-Mar-2018
`Valid Core License Keys:` issued after 01-Oct-2016.

- Pronto: Pronto! HTML Version 6.2.2 is included.
- MAPI: the MAPI Connector version 1.54.12.23 is included.
- New platform: Linux for Baikal-type processors (the `mipsel` architecture).
- SMTP: the SMTPUTF8 extension (RFC6531) implemented.
- ACCOUNTS: the "'From' Name Restrictions" option processing extended.
- XIMSS: event organizer is set to the shared calendar owner if the user has the Delegation right for the calendar owner account.
- XIMSS: replying to items in shared folders sets the From address to folder owner account if the Delegation right was granted.
- MAILBOX: the Delegation right on an account grants read-write access to folders in that account.
- PBX: added support for `Remote-Party-Id` header in `ProvisionCall`.
- CLI: the REPORTFAILEDLOGINADDRESS command has been implemented.
- KERNEL: the `--randomDataDevice` startup parameter is implemented for the path to random seed data device.
- WEBADMIN: cache controls of the paged output of LDAP search results implemented.
- HTTP: 'autodiscover' processing for Outlook 2016 has been improved.
- MAPI: Directory search used to build GAL now traverses subtrees by default.
- Bug Fix: CalDAV: calendar items might lack the From header.
- Bug Fix: TLS: 6.1.6: closed the possible vulnerability to ROBOT attacks using "bad-

version oracle" vector.
- Bug Fix: AIRSYNC: 6.2c5: meeting invitations without attendees might block synchronization of some clients.
- Bug Fix: AIRSYNC: 5.2: server might crash while parsing incorrect calendar data.
- Bug Fix: AIRSYNC: 5.3: message flags for some message types might fail to synchronize to client device.
- Bug Fix: AIRSYNC: 5.4.1: updated calendar items might lose description text.
- Bug Fix: WEBSKIN: addressed leak of session info through the usage of some scripts in the basic skin.
- Bug fix: XIMSS: ICE Candidate attribute was not set in XML SDP data.

**6.2.1** 26-Dec-2017
`Valid Core License Keys:` issued after 01-Oct-2016.

- Pronto: [Pronto! HTML Version 6.2.1](#) is included.
- MAPI: the [MAPI Connector version 1.54.12.22](#) is included.
- XIMSS: the `callHelper` operation has been implemented.
- SIGNAL: the `Map Call Dialog Status to Presence` account setting has been implemented.
- Bug Fix: QUEUERULES: 5.4: server might crash on incorrect rules data.
- Bug Fix: CHRONOS: 6.2.0: server might crash while processing meeting reminders.
- Bug fix: SIGNAL: wrong media IP for channels looped through external proxies.
- Bug fix: SMTP: 6.2c1: DMARC verification could reject messages from authenticated senders.

**6.2.0** 20-Nov-2017
`Valid Core License Keys:` issued after 01-Oct-2016.

- IMPORTANT: the End User License Agreement has been modified.
- Pronto: [Pronto! HTML Version 6.2.0](#) is included.
- PKI: added support for Subject Alternative Names in certificate requests and self-signed certificates.
- KERNEL: the initial support for encrypted mailboxes has been implemented.
- QUEUERULES: the `Calendar Method` condition for account and domain level rules has been implemented.
- QUEUERULES: the `whitelisted` option for the Source condition has been implemented.
- KERNEL: the Application Helpers has been implemented.
- CG/PL: the `CallHelper` function has been implemented.
- Bug Fix: AIRSYNC: 6.2c5: meetings might be duplicated.
- Bug Fix: AIRSYNC: 5.2: synchronizing large folder trees with large messages might cause server crashes.
- Bug Fix: LIST: 6.1.5: incorrect encoding of specific distribution subjects could break message formatting.

**6.2c5** 31-Oct-2017
`Valid Core License Keys:` issued after 01-Oct-2016.

- Pronto: [Pronto! HTML Version 6.2c5](#) is included.
- MAPI: the [MAPI Connector version 1.54.12.21](#) is included.
- WEBADMIN: the Threads monitor now displays CPU utilization values per thread (for Windows, Linux and FreeBSD 11).
- ACCOUNTS: the `Reroute External Recipients to` option has been implemented.
- ACCOUNTS: the initial support for application passwords has been implemented.
- CLI: the GETACCOUNTONESETTING command has been implemented.
- PKI: the built-in Trusted Root Certificates list has been renewed.
- WEBADMIN: monitoring Failed Logins and Protocol Errors IP Address lists has been implemented.
- WEBADMIN: Urgent Notifications have been implemented.
- Bug Fix: CHRONOS: 5.4: unprocessed schedules might be discarded on server stop.

- Bug Fix: AIRSYNC: 5.2: particular events in recurring series could not be declined.
- Bug Fix: AIRSYNC: 5.2: additional attendees in event exceptions were not processed.
- Bug Fix: AIRSYNC: 5.2: meeting requests and responses might be not processed properly on iOS devices.
- Bug Fix: AIRSYNC: 5.3: logging responses at All Info might crash server.
- Bug Fix: LIST: 5.4: long real names could be encoded incorrectly.
- Bug fix: SIP: 6.1: the "Optional Media security" option was not properly processed.
- Bug Fix: ACCOUNTS: 5.1: multiple simultaneous failures to login into the same account could crash the server.

**6.2c4** 21-Sep-2017
`Valid Core License Keys:` issued after 01-Sep-2016.

- Pronto: Pronto! HTML Version 6.2c4 is included.
- MAPI: the MAPI Connector version 1.54.12.20 is included.
- LIST: the `Hide 'From' Addresses` setting is replaced with `Compose 'From' Address as:`.
- ACCOUNTS: added option to require logins over encrypted connections.
- WEBSKIN: extra protection from WebUser session stealing and cross-site scripting.
- WEBUSER: extra protection from WebUser session stealing and cross-site scripting.
- UTILITIES: the `mail` utility now supports `-E` parameter.
- UTILITIES: the `sendmail` utility now ignores more parameters for compatibility.
- LDAP: the paged output (RFC2696) of search results now works faster.
- Kernel: `SHA-384` and `SHA-512` digesting algorithms have been implemented.
- WEBADMIN: monitoring Temporary Blocked and Temporary Client IP Address lists implemented.
- WEBADMIN: In Mail-SMTP-Receiving monitor the authenticated account name is displayed now.
- WEBADMIN: In Access monitors the TLS-encrypted connections are indicated now.
- XIMSS: the `calendarForward` operation has been implemented.
- ACCOUNTS: the `Authentication URI` option has been implemented (for LDAP).
- ROUTER: the `_dir` suffix support has been implemented for Directory-assisted relaying.
- HTTP: added support for multiple Set-Cookie fields in responses to requests.
- Bug Fix: DOMAINS: 6.2c3: Messages from mailing lists were not signed with DKIM.
- Bug Fix: TLS: 6.2c2: AES GCM encryption could crash the server on some platforms.
- Bug Fix: SMTP: some locally generated messages might be considered as relayed and blocked with restricted relaying settings.
- Bug Fix: HTTP: some requests with `chunked` encoding proxied to cluster backends might be processed incorrectly.
- Bug Fix: Directory: when deleting a Local Unit some files could stay orphaned.
- Bug Fix: AIRSYNC: 5.2: calendar items updated server-side could not be removed.

**6.2c3** 29-Apr-2017
`Valid Core License Keys:` issued after 01-Apr-2016.

- Pronto: Pronto! HTML Version 6.2c3 is included.
- WEBUSER: editing multiple Identities implemented.
- CLI: The VERIFYACCOUNTIDENTITY method implemented.
- ACCOUNTS: `'From' Address Restrictions` and `'From' Name Restrictions` settings implemented.
- QUEUE: verifying the validity of From: header in messages from authenticated senders implemented.
- SIGNAL: Push notifications to iOS and Android devices for incoming calls and IMs have been implemented.
- LDAP: search for non-routable address under the `dc=cgprouter` base now returns empty result rather than routing error.
- XMPP: blacklisting by domain name has been implemented.
- XIMSS: notifications on mailbox changes are sent now also when a mailbox is created/renamed/removed.
- QUEUERULES: the actions `Accept Request` and `Accept Reply` can be applied now to Tasks Requests and Replies.
- Bug Fix: AIRSYNC: 5.4: double quotes in the display name part of email addresses might be processed incorrectly.

- Bug Fix: LIST: 4.1: header/trailer texts were added with extra EOLs to HTML-formatted messages.

**6.2c2** 31-Mar-2017
`Valid Core License Keys`: issued after 01-Mar-2016.

- Pronto: [Pronto! HTML Version 6.2c2](#) is included.
- DOMAINS: adding DKIM Signatures to outgoing messages implemented.
- PBXLEG: `SendCallRegister` function is documented.
- RSIP: registration period is properly adjusted according to remote server responses.
- SIP: the "system overloaded" is now mapped to SIP error code 503.
- SIP: the RetryAfter parameter in SIP 503 responses is respected now.
- SIP: implemented the option to log call-related SIP requests and responses at the All Info level.
- SMTP: domain name in server prompt now follows the same rules as parameter to `HELO/EHLO`.
- SMTP: certificates from secondary domains can be sent when requested by the remote server.
- TLS: AES Galois Counter Mode cipher suites are implemented.
- QUEUE: Message revocation mechanism has been implemented.
- Bug Fix: PBX: 5.4: error code 423 was not handled correctly.
- Bug Fix: AirSync: 5.2: extra quotes might be added to real names in e-mail addresses and break formatting.
- Bug Fix: TLS: 6.1c processing client certificates could not be optional.

**6.2c1** 14-Feb-2017
`Valid Core License Keys`: issued after 01-Feb-2016.

- WEBADMIN: the "with DMARC" item added to "Check SPF records:" menu in SMTP module settings.
- SMTP: RFC7489 (Domain-based Message Authentication, Reporting, and Conformance (DMARC)) implemented (the authentication part).
- Kernel: RFC6376 (DomainKeys Identified Mail (DKIM) Signatures) signature checking implemented as part of DMARC support.
- Kernel: the Public Suffix List of domains is included as a service (required by DMARC).
- WEBUSER: now only folders of appropriate class can be selected for specific purposes (default Calendar, Tasks etc.).
- Kernel: the account preferences data were moved from account.info file into a separate file.
- DOMAINS: the Comment attribute is implemented.
- WEBADMIN: the Scheduled Tasks of an Account are now visible in Status page.
- ACCOUNTS: the File Storage now supports files larger than 4GB.
- FTP: the module now supports transfer of files larger than 4GB.
- WEBUSER: the "Redirect all Mail to" simplified rule redesigned to be DMARC-compatible and renamed to "Copy all Mail to".
- WEBUSER: choosing From: address from multiple Identities implemented.
- WEBUSER: default Directory Address Books implemented.
- STATISTICS: the `cpuTimeUsed` element is implemented.
- ACCOUNTS: Password expiration option is implemented.
- WEBADMIN: Limitation and Rate Control for the number of recipients in outgoing e-mail messages are implemented.
- WEBADMIN: Supplementary Log Files now can be deleted automatically.
- WEBADMIN: the Log Out button has been added.
- CG/PL: the `ReadSkinObject` function has been documented.
- SESSION: two-factor authentication framework has been implemented.
- XIMSS: the protocol has been extended to support two-factor authentication and forced password change.
- WebUser: the interface has been extended to support two-factor authentication and forced password change.

# Summary

## ACCOUNTS

- Password expiration option is implemented.
- The File Storage now supports files larger than 4GB.
- Account stores the last five IP addresses confirmed with Two-Factor Authentication.
- `'From' Address Restrictions` and `'From' Name Restrictions` settings implemented.
- Added option to treat only logins over encrypted connections as secure.
- The `Authentication URI` option has been implemented (for LDAP).
- The `Reroute External Recipients to` option has been implemented.

## AIRSYNC

- Multiple fixes to problems with calendar event processing.

## CG/PL

- The `ReadSkinObject` function has been documented.
- The `SendCallRegister` function has been documented.
- The `CallHelper` function has been implemented.

## CLI

- The BLESSSESSION method has been implemented.
- The VERIFYACCOUNTIDENTITY method has been implemented.
- The GETACCOUNTONESETTING command has been implemented.
- The REPORTFAILEDLOGINADDRESS command has been implemented.

## DOMAINS

- The Comment attribute is implemented.
- Adding DKIM Signatures to outgoing messages implemented.

## FTP

- The module now supports transfer of files larger than 4GB.

## HTTP

- 'autodiscover' processing for Outlook 2016 has been improved.

## KERNEL

- The support for encrypted mailboxes has been implemented.
- The Application Helpers has been implemented.
- The account preferences data were moved from account.info file into a separate file.
- The Public Suffix List of domains is included as a service (required by DMARC).
- RFC6376 (DomainKeys Identified Mail (DKIM) Signatures) implemented.
- `SHA-384` and `SHA-512` digesting algorithms have been implemented.

## LDAP

- Search for non-routable address under the `dc=cgprouter` base now returns empty result rather than routing error.
- The paged output (RFC2696) of search results now works faster.

## LIST

- The `Hide 'From' Addresses` setting is replaced with `Compose 'From' Address as:`

## MAILBOX

- The Sliced format is used by default now.
- Encrypted variants of mailbox formats are supported now.

## PKI

- The built-in Trusted Root Certificates list is renewed.
- Added support for Subject Alternative Names in certificate requests and self-signed certificates.

## PRONTO

- [Pronto! HTML Version 6.2](#) is included. It is the default "Pronto" interface now.

## QUEUE

- Message revocation mechanism has been implemented.
- Verifying the validity of From: header in messages from authenticated senders implemented.

## QUEUERULES

- The actions `Accept Request` and `Accept Reply` can be applied now to Tasks Requests and Replies.
- The Calendar Method condition for account and domain level rules has been implemented.
- The `whitelisted` option for the Source condition has been implemented.

## ROUTER

- The `_dir` suffix support has been implemented for Directory-assisted relaying.

## RSIP

- Registration period is properly adjusted according to remote server responses.

## SIGNAL

- Push notifications to iOS and Android devices for incoming calls and IMs have been implemented.
- The `Map Call Dialog Status to Presence` account setting has been implemented.

## SIP

- The "system overloaded" is now mapped to SIP error code 503.
- The RetryAfter parameter in SIP 503 responses is respected now.
- Implemented the option to log call-related SIP requests and responses at the All Info level.

## SMTP

- RFC7489 (Domain-based Message Authentication, Reporting, and Conformance (DMARC)) implemented (the authentication part).
- Domain name in server prompt now follows the same rules as parameter to `HELO/EHLO`.
- Certificates from secondary domains can be sent when requested by the remote server.
- The SMTPUTF8 extension (RFC6531) implemented.

## STATISTICS

- The `cpuTimeUsed` element is implemented.

## TLS

- AES Galois Counter Mode cipher suites are implemented.

## WEBADMIN

- Supplementary Log Files now can be deleted automatically.
- Limitation and Rate Control for the number of recipients in outgoing e-mail messages are implemented.
- The Scheduled Tasks of an Account are now visible in Status page.
- The "with DMARC" item added to "Check SPF records:" menu in SMTP module settings.
- The Log Out button has been added.
- Monitoring Temporary Blocked, Failed Logins, Protocol Errors, and temporary Client IP Address lists implemented.
- In Mail-SMTP-Receiving monitor the authenticated account name is displayed now.
- In Access monitors the TLS-encrypted connections are indicated now.
- The Threads monitor now displays CPU utilization values per thread (for Windows, Linux and FreeBSD 11).
- Urgent Notifications implemented.
- Cache controls of the paged output of LDAP search results implemented.

## WEBUSER

- Default Directory Address Books are implemented.
- The "Redirect all Mail to" simplified rule has been redesigned to be DMARC-compatible and renamed to "Copy all Mail to".
- Choosing From: address from multiple Identities implemented.
- Now only Calendar-type mailbox can be selected as Main Calendar, only Tasks-type one as Main Tasks, etc.
- Two-factor authentication is supported now.
- Editing Identities has been implemented.
- Extra protection from session hijacking has been implemented.

## XIMSS

- Commands for the Two-factor authentication is supported now.
- Notifications on mailbox changes are sent now also when a mailbox is created/renamed/removed.
- The `calendarForward` operation has been implemented.
- The `callHelper` operation has been implemented.

## XMPP

- Blacklisting by domain name has been implemented.

## UTILITIES

- The `mail` utility now supports `-E` parameter.
- The `sendmail` utility now ignores more parameters for compatibility.

---

CommuniGate Pro

| Introduction | Features | History | How To | Help Me |
|---|---|---|---|---|

# Version 6.1 Revision History

- 6.2
- 6.1: Summary
- 6.0
- 5.4
- 5.3
- 5.2
- 5.1
- 5.0
- 4.x
- 3.x
- 2.x
- 1.x

- Pronto!
- MAPI Connector

RSS

**6.1.19** 26-Dec-2017
`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: Pronto! HTML Version 6.1.19 is included.
- MAPI: the MAPI Connector version 1.54.12.22 is included.
- WEBUSER: extra protection from WebUser session stealing and cross-site scripting.
- WEBSKIN: Crystal: fixed scripting vulnerability.
- Bug Fix: AIRSYNC: 5.2: calendar items updated server-side could not be removed.
- Bug Fix: QUEUERULES: 5.4: server might crash on incorrect rules data.

**6.1.18** 15-Sep-2017
`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: Pronto! HTML Version 6.1.18 is included.
- MAPI: the MAPI Connector version 1.54.12.20 is included.
- WEBUSER: extra protection from WebUser session stealing and cross-site scripting.
- KERNEL: the logic of searches through host lists changed to stop on the first match.
- Bug Fix: WEBUSER: 6.1.17: some messages could not be rendered in framed views.
- Bug Fix: AIRSYNC: 6.1.8: declined calendar events stayed in calendar without notifications to the event organizer.

**6.1.17** 25-Aug-2017
`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: Pronto! HTML Version 6.1.17 is included.
- MAPI: the MAPI Connector version 1.54.12.18 is included.
- WEBSKIN: extra protection from WebUser session stealing and cross-site scripting.

WEBUSER: extra protection from WebUser session stealing and cross-site scripting.
- CALENDAR: updates to recurring items are processed even if original item is missing.
- TLS 1.2 compatibility with other implementations has been improved.
- SMTP: attacker IP is blacklisted also when errors are made in separate sessions.
- AIRSYNC: added workaround for task updates that miss task start time.
- Bug Fix: SMTP: some locally generated messages might be considered as relayed and blocked with restricted relaying settings.
- Bug Fix: HTTP: some requests with `chunked` encoding proxied to cluster backends might be processed incorrectly.
- Bug Fix: TLS: the server might crash on TLS handshake with incorrect elliptic curves parameters.

**6.1.16** 31-Mar-2017
`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: [Pronto! HTML Version 6.1.16](#) is included.
- Default password complexity is now enforced during batch account import.
- Bug Fix: TLS: 6.1c processing client certificates could not be optional.
- Bug Fix: WebUser: 6.1.10: server might crash on some requests if postmaster account was not available.
- Bug Fix: SIP: 6.1.14: incorrect SDP offer might be generated in some call scenarios.
- Bug Fix: AirSync: 5.2: extra quotes might be added to real names in e-mail addresses and break formatting.

**6.1.15** 14-Feb-2017
`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: [Pronto! HTML Version 6.1.15](#) is included.
- MEDIASERVER: peer detection mechanism has been improved.
- Bug Fix: LDAP: 5.4: extended response packets were constructed incorrectly.
- Bug Fix: SIP: 6.1c1: the `FixContact` workaround was processed incorrectly.
- Bug Fix: SIP: 6.1c1: the `noPath` workaround was processed incorrectly.
- Bug Fix: MEDIAPROXY: 6.1c1: SRTP-decoding media proxy might fail under high load.

**6.1.14** 28-Dec-2016
`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: [Pronto! HTML Version 6.1.14](#) is included.
- IMAP: the support for RFC6851 is disabled by default.
- SMTP: domains with MX pointing to server's IPs and not routed within the server are not accepted for relaying.
- CLI: password complexity checks are skipped only when administrator has rights to change complexity.
- WebAdmin: password complexity checks are skipped only when administrator has rights to change complexity.
- IM: the option to deliver all incoming IMs to all sessions and devices has been implemented.
- MEDIAPROXY: peer detection mechanism has been improved.
- Bug Fix: 6.1.9: Foundation: conversion of multi-byte charsets to utf-8 could cause memory corruption.
- Bug Fix: 5.3: Core: authentication using client certificates might fail.
- Bug Fix: 6.0.4: Rules: some non-ASCII encoded attachment names might be decoded incorrectly.

**6.1.13** 21-Nov-2016
`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: [Pronto! HTML Version 6.1.13](#) is included.
- MAPI: the [MAPI Connector version 1.54.12.11](#) is included.
- CG/PL: the syntax of the `CryDigest` method has been extended to support `HMAC` algorithm.
- WebAdmin: fixes to the Dash skin to improve usability.
- Bug Fix: 6.1.12: WebUser: long list of addresses in To/Cc/Bcc fields might be truncated.

**6.1.12** 30-Sep-2016

`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: [Pronto! HTML Version 6.1.12](#) is included.
- MAPI: the [MAPI Connector version 1.54.12.9](#) is included.
- PKI: the option to select signature algorithm for SSL/TLS certificates and signing requests has been implemented.
- WebUser: handling of the `mailto:` links has been improved.
- DTLS: workaround for Chrome WebRTC / OpenSSL DTLS bugs is implemented.
- IMAP: the support for RFC6851 has been documented and enabled.
- PLATFORM: AIX: PowerPC 64 bit is supported with a native build now.
- Bug Fix: 5.0c1: SIP: the alternative "l" header field was not supported for packets received via TCP/TLS.
- Bug Fix: 6.1c1: PBXLEG: server might crash on a bridged call attempt with all media ports exhausted.
- Bug Fix: 6.1c1: MEDIASERVER: cluster IP might be exposed to external peers in cases of intra-cluster bridging.
- Bug Fix: 6.0.1: LDAP: some requests under the `dc=cgprouter` subtree might be processed incorrectly.
- Bug Fix: 6.1c4: FTP: using invalid paths with some commands might cause server crash.
- Bug Fix: 6.1c1: WebSkins: Crystal: address books might fail to list records.
- Bug Fix: 6.1c1: PKI: self-generated S/MIME issuer certificate lacked critical basic constrains.

**6.1.11** 20-Jun-2016

`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: [Pronto! HTML Version 6.1.11](#) is included.
- MAPI: the [MAPI Connector version 1.54.12.7](#) is included.
- EULA has been updated.
- STUN: unknown attributes cause errors only if they are in the "required" category.
- CardDAV: absolute references are supported in address book "multiget" requests.
- WebUser: the Crystal skin implements the support for identities.
- Bug Fix: 6.1c1: PBXLEG: some bridging scenarios might fail.
- Bug Fix: 6.1c1: CGI: incorrect format was used for the REMOTE_ADDR CGI variable.
- Bug Fix: 6.1c4: CG/PL: some complex CG/PL programs might fail to compile.
- Bug Fix: 6.0: IM: storing of IMs received while offline could not be turned off.
- Bug Fix: 6.1.9: AIRSYNC: initial sync might fail for some clients.
- Bug Fix: 6.1.10: QUEUERULES: the Vacation rule start date was processed incorrectly.
- Bug Fix: 5.0: CALENDAR: some complex time zone definitions were processed incorrectly.
- Bug fix: 5.x: XIMSS: non-routable addresses blocked remaining valid addresses from being included into replies.
- Bug Fix: 5.x: CLUSTER: some specific operations on shared folders might cause server crash.

**6.1.10** 20-Apr-2016

`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: [Pronto! HTML Version 6.1.10](#) is included.
- MAPI: the [MAPI Connector version 1.54.12.6](#) is included.
- QUEUERULES: the Vacation rule functionality is extended.
- IMAP: the `XLIST` command is implemented.
- CLI: the syntax of the RENAMEDOMAIN command has been extended to allow storage point specification.
- CLUSTER: the remote server connection state is checked before a Virtual Mailbox is reused.
- PBXApp: the RejectCall parameters are extended.
- SIGNAL: when a forked signal cannot be cancelled, it is immediately removed from the fork set.
- Bug Fix: 6.1.8: Calendar: IM and E-Mail calendar alarms might show the time till event incorrectly.
- Bug Fix: 6.0: AIRSYNC: mailbox search parameters were processed incorrectly.
- Bug Fix: 6.1.7: CalDAV: calendar parsing might loop on requests with incomplete filter data.
- Bug Fix: 6.1c2: XIMSS: Notifications for mailboxes with non-ASCII names might be lost.

- Bug Fix: 6.1c1: CALLLEG: a response to an INVITE/replaces request could drop the MediaProxy built for the request.
- Bug Fix: 6.1c1: CALLLEG: timeout, CANCEL and incoming BYE in the PRACK-waiting state could be processed incorrectly.
- Bug Fix: 6.1.3: PBXLEG: server might crash on bridged calls with non-AV media streams.
- Bug Fix: 6.1c2: PBXLEG: update soliciting could fail.
- Bug Fix: 6.1.5: Core: S/MIME signatures for attachments with non-ASCII names might be calculated incorrectly.
- Bug Fix: 4.0: SNMP: the GetNext request for a non-array "type" object did not return the "leaf" object.

**6.1.9** 02-Mar-2016
`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: Pronto! HTML Version 6.1.9 is included.
- AIRSYNC: the support for availability data retrieval has been implemented.
- CORE: added support for UTF-7 encoding.
- XIMSS: added workaround for incorrectly encoded attachment names.
- CLI: the syntax of the RENAMEACCOUNT command has been extended to allow storage point specification.
- SIP: implemented domain prefixes for registration of endpoints that require secure media.
- Bug Fix: 6.1: DIALOG: MEDIAPROXY short-circuit optimization could result in media absence in transferred calls.
- Bug Fix: 5.4: CLUSTER: failed WebUser logins in cluster environments might be used for valid account name discovery.
- Bug Fix: 6.1c2: AIRSYNC: handling some requests might lead to memory corruption.
- Bug Fix: 6.1c1: DIALOG: non-fatal negative mid-call response could close the call.
- Bug Fix: 6.1c1: QUEUE: server might crash on mail queuing to non-email modules.
- Bug Fix: 6.0c1: LOCAL: random memory corruption and crashes might happen under a heavy load.
- Bug Fix: 5.4: ACCOUNT: file lock owner information could be stored without XML namespaces.

**6.1.8** 30-Dec-2015
`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: Pronto! HTML Version 6.1.8 is included.
- Media: workaround for some WebRTC clients that depend on SDP attributes order.
- Calendar: the text of IM and E-Mail reminders has been improved.
- AIRSYNC: server-side replies to meeting invitations are suppressed in favour to those generated by clients themselves.
- Bug Fix: 6.1c2: XIMSS: server might crash on events sent to XIMSS sessions.
- Bug Fix: 6.1.7: PLATFORM: MacOS X installation scripts attempted to rename the `mail` utility with the "rootless mode" in effect.

**6.1.7** 05-Nov-2015
`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: Pronto! HTML Version 6.1.7 is included.
- Kernel: Elliptic Curve Private Key support is implemented (RFC5915, RFC4492, RFC5289).
- TLS: additional cipher suites are implemented.
- AIRSYNC: order of properties in the ItemOperations/Fetch request result changed to help some clients with attachments downloading.
- PLATFORM: MacOS X: the location of the Application Directory changed from `/usr/sbin` to `/usr/local` to meet the new security requirements.
- Bug Fix: 5.4: SMPP: incorrect NPI value was used for alpha-numeric TON.
- Bug Fix: 5.3: AIRSYNC: meeting attendees received duplicated invitations and updates.
- Bug Fix: 5.3: CalDAV: item filters might be applied incorrectly so no events were shown in some clients.

Bug Fix: DOMAINS: all domain's directory records were deleted with domain affecting distributed domains.
- Bug Fix: 6.1c3: WebUser: Calendars could not be scrolled to previous dates.
- Bug Fix: 6.1.3: Directory: routing of signal requests in some distributed domain configurations might fail.
- Bug Fix: 6.1.6: WebSkin: rich text formatting in message composer was broken.
- Bug Fix: 6.0: Core: converting text to two-bytes character sets might crash the server.

**6.1.6** 30-Sep-2015
`Valid Core License Keys:` issued after 01-Feb-2014.

- Pronto: [Pronto! HTML Version 6.1.6](#) is included.
- MAPI: the [MAPI Connector version 1.54.12.5](#) is included.
- Kernel: Diffie-Hellman PKI cryptography is implemented.
- Kernel: Elliptic Curves cryptography is implemented.
- TLS: Ephemeral Diffie-Hellman suites are implemented.
- TLS: Ephemeral Elliptic Curves suites are implemented.
- TLS: PFS (Perfect Forward Secrecy) is implemented.
- Foundation: enhanced upsampling/downsampling algorithms for G.711 codecs are implemented.
- WebSkin: Simplex: improved JS code to detect the MS Edge browser.
- CalDAV: added workaround for clients that fail to use parent event when rendering exceptions.
- Bug Fix: 6.0: AIRSYNC: calendar item was not updated on the server when only end time was changed.
- Bug Fix: 6.1.1: Media: some SRTP-related attributes might be incorrectly output to SDP text.
- Bug Fix: 6.1.3: Core: MIME atoms with multi-byte encodings might be wrapped incorrectly.
- Bug Fix: 6.1.5: Core: long names of attached files might be encoded incorrectly.
- Bug Fix: 6.c1: SIGNAL: A/V calls could be forked to XMPP sessions.

**6.1.5** 3-Aug-2015
`Valid Core License Keys:` issued after 01-Feb-2014.

- CalDAV: Meeting invitations are sent automatically.
- TLS: with Weak Ciphers disabled RC4 cipher is advertised only for old clients (SSL 3.0 and TLS 1.0).
- Pronto: [Pronto! HTML Version 6.1.5](#) is included.
- Bug Fix: 6.1c1: WebSkins: Crystal: adding attachments to messages did not work in Internet Explorer 11.
- Bug Fix: 6.1c1: Core: some non-ASCII names of attached files might be encoded incorrectly.
- Bug Fix: 5.4: XMPP: the 'from' attribute was not included into server-server responses.

**6.1.4** 25-Jun-2015
`Valid Core License Keys:` issued after 01-Feb-2014.

- AIRSYNC: workaround for possible E-Mail address corruption in Contacts by some Android devices.
- Pronto: [Pronto! HTML Version 6.1.4](#) is included.
- Bug Fix: 6.1c4: Media: the support for peer-to-peer media might be improperly enabled.
- Bug Fix: 6.1c5: Mailbox: empty Sliced mailboxes might be reported in the logs as having damaged index file.
- Bug Fix: 6.1.3: HTTPU: some sub-protocol HTTP connections could not be restricted to Client IPs only.
- Bug Fix: 6.1c1: CLI: some read commands were not available to administrators with the Read All Domains right.

**6.1.3** 12-Jun-2015
`Valid Core License Keys:` issued after 01-Feb-2014.

- WebUser: "regular" login operations now check the HTTP WebUser Access setting, too.
- HTTP: when a CGI application is started, countermeasures for CVE-2014-6271 "bash" vulnerability are applied.
- Directory: routing of signal requests in distributed domains is improved.
- Pronto: [Pronto! HTML Version 6.1.3](#) is included.
- MAPI: the [MAPI Connector version 1.54.12.2](#) is included.
- Bug Fix: 6.1c2: XIMSS: certain TCP-based login methods could cause server crashes.

- Bug Fix: 6.1c1: SIP: CANCEL requests in a Cluster environment could be sent via a wrong cluster member.
- Bug Fix: 6.1.2: FTP: NLST command could crash the server.
- Bug Fix: Keys: plugin keys issued after April 2015 might be shown as expired.

**6.1.2** 16-Mar-2015
`Valid Core License Keys`: issued after 01-Feb-2014.

- Platform: Linux/ARMv6-11 (including Raspberry Pi) is supported now.
- SIP: now the "Paid SIP Server" setting is not applied to REGISTER requests.
- Pronto: Pronto! HTML Version 6.1.2 is included.
- Bug Fix: 6.1c1: CALLLEG: packets with misformed SDP could crash the server.
- Bug Fix: 6.1c1: MEDIASERVER: adjusting to client timers did not work properly.
- Bug Fix: 6.1c1: FTP: the NLST command could fail to enumerate directory contents.
- Bug Fix: 6.1c1: SIGNAL: the "presence on registration" SIP Workaround option could cause server crash.
- Bug Fix: 5.4: QUEUERULES: certain misformed operations could de-sync the cluster delivery protocol.
- Bug Fix: 2.0: MAILBOX: updating mailbox message flags by some session could reset internal "marked" flags set on that message by other sessions.
- Bug Fix: 5.4: MAILBOX: the "Erase deleted messages" function could fail for directory-type mailboxes.

**6.1.1** 23-Feb-2015
`Valid Core License Keys`: issued after 01-Feb-2014.

- MEDIAPROXY: peer detection mechanism has been improved.
- MEDIASERVER: peer detection mechanism has been improved.
- SIP: separate settings for the SRTP and DTLS media protocols are implemented.
- SIP: the Optional Media Security setting is implemented.
- Bug Fix: 6.1c2: XIMSS: tcp connections attempts from banned clients could crash the server.
- Bug Fix: 6.1c1: MEDIAPROXY: RTCP packets could be relayed to RTP ports.
- Bug Fix: 6.1c2: PBXLEG: the leg processor looped on answering start-bridged UPDATE request.
- Bug Fix: 6.0: some misformed RTP packets could crash the server.
- Bug Fix: 5.4: certain header fields could cause the server to loop while converting them into Chinese charsets.

**6.1** 15-Feb-2015
`Valid Core License Keys`: issued after 01-Feb-2014.

- ADMIN: the Can Read All Domain and Account Settings access right is implemented.
- ADMIN: the CanCreateSpecialAccounts Domain-level access right is implemented.
- ACCOUNTS: the Log Protocols setting is implemented.
- Pronto: Pronto! HTML Version 6.1 is included. It is the default "Pronto" interface now.
- Platform: Sparc Solaris versions are built for the Sparc v8 architecture now, to avoid slowdowns on newer hardware.
- ROUTER: the .via, .relay, .domain, .local, .here, .xmpp, .xmppq, .enum, .noenum special domain name suffix strings are renamed into ._via, ._relay, ._domain, ._local, ._here, ._xmpp, ._xmppq, ._enum, ._noenum suffix strings.
- MAILBOX: Sliced mailbox crash-recovery has been improved.
- DIAMETER: the DIAMETER module is decommissioned.
- Bug Fix: 6.1c1: ACCOUNT: limiting the number of concurrenly open Account sessions could crash the server.
- Bug Fix: 6.1c1: ACCOUNT: accounts in secondary domains scheduled for mail store re-scanning were rescanned in a loop.
- Bug Fix: 6.0c1: STATISTICS: the string-type trigger events were not restored after server restart.
- Bug Fix: 6.1c1: CLUSTER: the REJECTQUEUEMESSAGES command was not working in Cluster environments.

**6.1c4** 05-Feb-2015
`Valid Core License Keys:` issued after 01-Feb-2014.

- CG/PL: global constants are implemented.
- Platform: a workaround for MS Windows kernel bug implemented. Stopped external tasks could remain undetected if several tasks were started at the same time.
- FTP: command processing redesigned to remove file path length limitations.
- AIRSYNC: now the Enabled Mobile Devices settings can be specified using the Account Default Settings.
- AIRSYNC: now the Enabled Mobile Devices settings can include allowed and prohibited device IDs and device types.
- PBX: the "telephony tester" application is included.
- Bug Fix: 6.0c2: CLUSTER: the ID of the killed NamedTask was sent incorrectly.

**6.1c3** 29-Jan-2015
`Valid Core License Keys:` issued after 01-Jan-2014.

- CLUSTER: when a server is marked as 'non-ready', its unlocked accounts are released immediately.
- WebUser: the Event and Task Categories fields can be modified now.
- WSSP: the REQUESTSOURCEIP function is implemented.
- PKI: certificates with SHA256 signatures are supported now.
- CLI: the TEMPBLACKLISTIP command is implemented.
- SMTP: the HELO Domain Prefix module setting is removed; the HELO Prefix setting is added to Domain SMTP settings.
- QUEUERULES: the Copy Attachments action is applied to one-part message containing an attachment.
- Bug Fix: SMTP: 6.1c1: the Require STARTTLS setting did not work.
- Bug Fix: HTTP: 6.1c1: cluster authentication could crash the server.
- Bug Fix: TLS: 6.1c1: communicating with products that generate empty data-blocks (such as those based on OpenSSL) could be interrupted.
- Bug Fix: CLUSTER: 6.0c2: if a signal is redirected to a Named Task, and that Task runs on a different member, the relayed request contained the original request URI.

**6.1c2** 12-Dec-2014
`Valid Core License Keys:` issued after 01-Nov-2013.

- WEBDAV: the "include" PROPFIND element is supported now.
- FILEDAV: directories now report the "httpd/unix-directory" value for the "getcontenttype" property.
- PBXApp: the "allowedAttributes","allowedSSRC" call parameters are implemented.
- XIMSS: the callRedirect operation supports the "fork" option now.
- AIRSYNC: the ValidateCertificate command is implemented.
- WebUser/Pronto: the "show images using proxy" option and the "show HTML in a frame" option can work together now.
- SMTP,POP,IMAP,FTP,HTTP: attempts to start SSL/TLS connections on clear text ports are detected now; connections are dropped immediately.
- LIST: the ^C symbol combination is supported in Feed headers and trailers.
- CLI: the CREATEDOMAINSTORAGE, LISTDOMAINSTORAGE, CREATEACCOUNTSTORAGE, LISTACCOUNTSTORAGE commands are implemented.
- DIRECTORY: when a Remote Unit reads a "search reference" record, the record is ignored, but the search continues.
- CG/PL: the MailboxAppend operation parameters have been changed.
- XIMSS: the messageAppend, contactAppend operation parameters have been changed.
- Bug Fix: DIALOG: 6.1c1: requests on "early" dialogs (UPDATE, INFO) could fail.
- Bug Fix: WEBADMIN: 6.1c1: account creation could fail when no "Mount Storage Point" was specified.
- Bug Fix: MEDIASERVER: 6.1c1: the GETPLAYPOSITION function did not return positive values.
- Bug Fix: MAPI: messages sent from the Main Domain were not marked as 'trusted'.

**6.1c1** 28-Nov-2014
`Valid Core License Keys`: issued after 01-Nov-2013.

- Pronto: [Pronto! HTML Version 6.1](#) is included. It is the default "Pronto" interface now.
- CALLLEG: the internal engine has been redesigned.
- PBXLEG: the module has been redesign to provide for correct offer/answer negotiations.
- WEBADMIN: the "Dash" Skin has been added.
- TLS: DTLSv1.0 (RFC4347) and DTLSv1.2 (RFC6347) are implemented .
- TLS: RFC5705 is implemented.
- TLS: the Oldest Accepted setting is implemented.
- Kernel: Email address header field composing has been improved.
- NETWORK: the Re-usage Delay setting is implemented.
- NETWORK: DNR statistics elements are added.
- NETWORK: now the Debug IP functionality is applied to UDP-based protocols, too.
- SMTP: when a remote host accepts the ATRN command and sends messages back to the Server, these messages are processed as coming from a "trusted" (client IP) source.
- QUEUE: now Message envelopes store the authenticated sender's account name.
- QUEUE: the Reject All Sender's Messages WebAdmin command is implemented.
- CLI: the REJECTQUEUEMESSAGES command is implemented.
- CLI: the CREATEDOMAIN command is extended; the CREATESHAREDDOMAIN command is deprecated.
- DOMAINS: the Storage Mount Point directories are implemented.
- DOMAINS: the Require STARTTLS setting is implemented.
- ACCOUNTS: the Storage Mount Point directories are implemented.
- ACCOUNTS: the POP, IMAP, AirSync, XMPP, XIMSS, WebUser Session Limit settings are implemented.
- ACCOUNTS: the "verify" type records are added to the Login Logs.
- QUEUERULES: External Filter protocol version 4 is implemented.
- AIRSYNC: Notes synchronization is implemented.
- AIRSYNC: "partial" SYNC requests are supported now.
- AIRSYNC: attempts to submit duplicate Email messages are detected now.
- AIRSYNC: the Settings command implementation has been extended.
- LOCAL: INBOX opening is postponed till actual message delivery.
- SMTP: message queue names for "Hold" domains do not include the source domain names now.
- SMTP: the Require STARTTLS setting is implemented.
- SIGNAL: the AOR Limit and Fork Limit settings are implemented.
- SIGNAL: the downstream request (ACK) processing moved to asynchronous objects.
- SIGNAL: the DIALOG (Call) manager and proxy-building mechanisms are re-implemented.
- SIGNAL: the CDR records now use the tabulation (instead of the space) symbol as the field separator.
- XIMSS: the 'version' login parameter is implemented.
- XIMSS: the signaling (call management) operations have been re-designed, protocol has changed.
- XIMSS: the EmptyMailbox operation is implemented.
- XIMSS: the setSessionOption operation is implemented.
- XIMSS: the setTimeout operation is deprecated.
- XIMSS: WebRTC media support is implemented.
- XIMSS: PROFILE retrieval is implemented.
- CALLLEG: 422 error code processing has been improved.
- CALLLEG: the Advertise UPDATE and Send UPDATE settings are deprecated.
- DIRECTORY: the LDAP Unit Search Filter setting is implemented.
- SIP: downstream request and upstream response processing moved to SIPS and SIPC asynchronous objects.
- SIP: now a MediaProxy can be built for INVITE requests without an SDP body.
- SIP: the BadUPDATE workaround option is implemented.
- SIP: the Transport Settings page is implemented.
- RSIP, RPOP: "next record to process" selection algorithm has been improved.
- IMAP: RFC6154 has been implemented.

GROUPWARE: Russian timezones are updated to reflect 2014 changes.
- PBXLEG: the "Reason" header field of provisioning responses is copied between egress and ingress legs of bridged calls.
- CLUSTER: the sent SMTP ATRN commands use the Cluster name now.
- CLUSTER: virtual mailbox updates are detected automatically now.
- WSSP: the RequestResource() function is implemented.
- WebApp: the GetHTTPResource() CG/PL function is implemented.
- WEBUSER: the "Crystal" Skin has been added.
- PBXApp: SignalOption() functionality has been extended.
- Foundation: the SDP XML representation now includes "info" elements.
- MEDIASERVER: RFC5763, RFC5764 (DTLS-SRTP) are implemented.
- MEDIASERVER: WebRTC media support is implemented.
- MEDIASERVER: inbound DTMF recognition now works for 8KHz, 16KHz, and 24KHz internal buffer sampling rates.
- MEDIASERVER: current audio level calculations have been improved.
- MEDIAPROXY: RFC5763, RFC5764 (DTLS-SRTP) are implemented.
- MEDIAPROXY: WebRTC media support is implemented.

# Summary

**ACCOUNTS**

- the Storage Mount Point directories are implemented.
- the POP, IMAP, AirSync, XMPP, XIMSS, WebUser Session Limit settings are implemented.
- the Log Protocols setting is implemented.
- the "verify" type records are added to the Login Logs.

**ADMIN**

- the Can Read All Domain and Account Settings access right is implemented.
- the CanCreateSpecialAccounts Domain-level access right is implemented.

**AIRSYNC**

- Notes synchronization is implemented.
- "partial" SYNC requests are supported now.
- attempts to submit duplicate Email messages are detected now.
- the Settings command implementation has been extended.
- the ValidateCertificate command is implemented.
- the Enabled Mobile Devices settings can be specified using the Account Default Settings.
- the Enabled Mobile Devices settings can include allowed and prohibited device IDs and device types.
- order of properties in the ItemOperations/Fetch request result changed to help some clients with attachments downloading.
- the support for availability data retrieval has been implemented.

**CALDAV**

- Meeting invitations are sent automatically.

**CALLLEG**

- the internal engine has been redesigned.
- 422 error code processing has been improved.

- the Advertise UPDATE and Send UPDATE settings are deprecated.

## CARDDAV

## CG/PL

- the MailboxAppend operation parameters have been changed.
- global constants are implemented.

## CLI

- the REJECTQUEUEMESSAGES command is implemented.
- the CREATEDOMAIN command is extended; the CREATESHAREDDOMAIN command is deprecated.
- the CREATEDOMAINSTORAGE, LISTDOMAINSTORAGE, CREATEACCOUNTSTORAGE, LISTACCOUNTSTORAGE commands are implemented.
- the TEMPBLACKLISTIP command is implemented.
- RENAMEACCOUNT and RENAMEDOMAIN commands support storage path specification.

## CLUSTER

- the sent SMTP ATRN commands use the Cluster name now.
- virtual mailbox updates are detected automatically now.
- when a server is marked as 'non-ready', its unlocked accounts are released immediately.

## DIAMETER

- the DIAMETER module is decomissioned.

## DIRECTORY

- the LDAP Unit Search Filter setting is implemented.
- when a Remote Unit reads a "search reference" record, the record is ignored, but the search continues.

## DOMAINS

- the Storage Mount Point directories are implemented.
- the Require STARTTLS setting is implemented.

## FILEDAV

- directories now report the "httpd/unix-directory" value for the "getcontenttype" property.

## FOUNDATION

- the SDP XML representation now includes "info" elements.

## FTP

- attempts to start SSL/TLS connections on clear text ports are detected now; connections are dropped immediately.
- command processing redesigned to remove file path length limitations.

## GROUPWARE

- Russian timezones are updated to reflect 2014 changes.
- Text of IM and E-Mail reminders has been improved.

## HELPERS

### HTTP

- attempts to start SSL/TLS connections on clear text ports are detected now; connections are dropped immediately.

### IMAP

- the support for RFC6851 has been documented and enabled.
- RFC6154 has been implemented.
- attempts to start SSL/TLS connections on clear text ports are detected now; connections are dropped immediately.
- added support for the XLIST command.

### KERNEL

- Email address header field composing has been improved.
- Diffie-Hellman PKI cryptography is implemented.
- Elliptic Curves cryptography is implemented.
- Elliptic Curve Private Key support is implemented (RFC5915, RFC4492).
- Support for UTF-7 encoding.

### LIST

- the ^C symbol combination is supported in Feed headers and trailers.

### LOCAL

- INBOX opening is postponed till actual message delivery.

### MAILBOX

- Sliced mailbox crash-recovery has been improved.

### MEDIAPROXY

- RFC5763, RFC5764 (DTLS-SRTP) are implemented.
- WebRTC media support is implemented.
- peer detection mechanism has been improved.

### MEDIASERVER

- RFC5763, RFC5764 (DTLS-SRTP) are implemented.
- WebRTC media support is implemented.
- inbound DTMF recognition now works for 8KHz, 16KHz, and 24KHz internal buffer sampling rates.
- current audio level calculations have been improved.
- peer detection mechanism has been improved.

### NETWORK

- the Reusage Delay setting is implemented.
- DNR statistics elements are added.
- now the Debug IP functionality is applied to UDP-based protocols, too.

## PBX

- the "telephony tester" application is included.

## PBXAPP

- SignalOption() functionality has been extended.
- the "allowedAttributes","allowedSSRC" call parameters are implemented.

## PBXLEG

- the module has been redesign to provide for correct offer/answer negotiations.
- the "Reason" header field of provisioning responses is copied between egress and ingress legs of bridged calls.

## PKI

- certificates with SHA256 signatures are supported now.
- the option to select signature algorithm for SSL/TLS certificates and signing requests has been implemented.
-

## PLATFORM

- a workaround for MS Windows kernel bug implemented. Stopped external tasks could remain undetected if several tasks were started at the same time.
- Sparc Solaris versions are built for the Sparc v8 architecture now, to avoid slowdowns on newer hardware.
- AIX PowerPC 64 bit is supported with a native build now.
- the location of the Application Directory on MacOS X changed to meet the new security requirements.

## POP

- attempts to start SSL/TLS connections on clear text ports are detected now; connections are dropped immediately.

## PRONTO

- [Pronto! HTML Version 6.1](#) is included. It is the default "Pronto" interface now.

## QUEUE

- now Message envelopes store the authenticated sender's account name.
- the Reject All Sender's Messages WebAdmin command is implemented.

## QUEUERULES

- External Filter protocol version 4 is implemented.
- the Copy Attachments action is applied to one-part message containing an attachment.
- the functionality of the Vacation rule is extended.

## ROUTER

- the .via, .relay, .domain, .local, .here, .xmpp, .xmppq, .enum, .noenum special domain name suffix strings are renamed into ._via, ._relay, ._domain, ._local, ._here, ._xmpp, ._xmppq, ._enum, ._noenum suffix strings.

## RPOP

- "next record to process" selection algorithm has been improved.

## RSIP

- "next record to process" selection algorithm has been improved.

## SIGNAL

- the AOR Limit and Fork Limit settings are implemented.
- the downstream request (ACK) processing moved to asynchronous objects.
- the DIALOG (Call) manager and proxy-building mechanisms are re-implemented.
- the CDR records now use the tabulation (instead of the space) symbol as the field separator.

## SIP

- downstream request and upstream response processing moved to SIPS and SIPC asynchronous objects.
- now a MediaProxy can be built for INVITE requests without an SDP body.
- the BadUPDATE workaround option is implemented.
- the Transport Settings page is implemented.
- Domain prefixes for registration of endpoints that require secure media are implemented.

## SKINS

## SMPP

## SMTP

- when a remote host accepts the ATRN command and sends messages back to the Server, these messages are processed as coming from a "trusted" (client IP) source.
- message queue names for "Hold" domains do not include the source domain names now.
- the Require STARTTLS setting is implemented.
- attempts to start SSL/TLS connections on clear text ports are detected now; connections are dropped immediately.
- the HELO Domain Prefix module setting is removed; the HELO Prefix setting is added to Domain SMTP settings.

## STATISTICS

## TLS

- TLS: DTLSv1.0 (RFC4347) and DTLSv1.2 (RFC6347) are implemented .
- TLS: RFC5705 is implemented.
- TLS: the Oldest Accepted setting is implemented.
- TLS: Ephemeral Diffie-Hellman suites are implemented.
- TLS: Ephemeral Elliptic Curves suites are implemented.
- TLS: PFS (Perfect Forward Secrecy) is implemented.
- TLS: additional cipher suites are impemented.

## WEBADMIN

- the "Dash" Skin has been added.

## WEBAPP

- the GetHTTPResource() CG/PL function is implemented.

## WEBDAV

- the "include" PROPFIND element is supported now.

**WEBUSER**

- the "Crystal" Skin has been added.
- the "show images using proxy" option and the "show HTML in a frame" option can work together now.
- the Event and Task Categories fields can be modified now.

**WSSP**

- the RequestResource() function is implemented.
- the REQUESTSOURCEIP function is implemented.

**XIMSS**

- the 'version' login parameter is implemented.
- the signaling (call management) operations have been re-designed, protocol has changed.
- the EmptyMailbox operation is implemented.
- the setSessionOption operation is implemented.
- the setTimeout operation is deprecated.
- WebRTC media support is implemented.
- PROFILE retrieval is implemented.
- the callRedirect operation supports the "fork" option now.
- the messageAppend, contactAppend operation parameters have been changed.

**XMPP**

| Introduction | Features | **History** | How To | Help Me |
|---|---|---|---|---|

# Version 6.0 Revision History

- **6.2**
- **6.1**
- **6.0: Summary**
- **5.4**
- **5.3**
- **5.2**
- **5.1**
- **5.0**
- **4.x**
- **3.x**
- **2.x**
- **1.x**

- **Pronto!**
- **MAPI Connector**

RSS

Cluster upgrade note: for rolling upgrades, your Cluster must be upgraded to the 5.4.9 or better version before upgrading to any 6.0 version.

**6.0.11** 10-Dec-2014

`Valid Core License Keys:` issued after 01-Dec-2011.

- TLS: the server-side SSLv3 protocol is disabled by default.
- WebMail/Pronto: HTML links pointing to anchors in the same E-mail body part are supported now.
- Bug Fix: MEDIA: 5.1: if the codec sample rate was different from the internal buffer rate, recording was started with a delay.
- Bug Fix: MAILBOX: 5.0: message replacing did not check for 2GB size limit in Text-type mailboxes
- Bug Fix: SMTP: 5.4: outgoing SMTP connections for messages without a "source domain" assigned could be initiated from non-default IP addresses.

**6.0.10** 08-Oct-2014

`Valid Core License Keys:` issued after 01-Dec-2011.

- MAPI: the MAPI Connector version 1.54.12.1 is included.
- Pronto: Pronto! Version 6.0.10 and Pronto Lite 2.3 are included.
- WebUser: the default name for the ShiftJIS encoding is changed from "Shift-JIS" to "Shift_JIS".
- Bug Fix: XIMSS: 6.0.2: attended call transfers were not handled.
- Bug Fix: XIMSS: 6.0c1: the server might crash on the SMIMESet command used in incorrect context.
- Bug Fix: LDAP: the searches under the special `dc=cgprouter` subtree did not return all valid results.

Bug Fix: vCard: parsing bogus data might crash the server.
- Bug Fix: SIP: 5.3: some remote Contact URIs might be incorrectly treated as local, causing SIP dialog setup errors.
- Bug Fix: CGPL: 5.2: ToLowerCase()/ToUpperCase() methods processed only the first symbol in the parameter string.
- Bug Fix: MailboxBSD: 5.4: under heavy load, a mailbox could be parsed multiple times, causing index deletion.

**6.0.9** 31-Mar-2014
`Valid Core License Keys`: issued after 01-Dec-2011.

- Pronto: [Pronto! Version 6.0.9 and Pronto Lite 2.2](#) are included.
- XIMSS: The "Keep time in Trash" preference is used now to preserve timestamps of items moved to Trash folder
- Bug Fix: XIMSS: 6.0c1: the server might crash on the SMIMESet command with incorrect parameters.
- Bug Fix: CALENDAR: 5.4: the FreeBusy data might be calculated incorrectly.
- Bug Fix: TLS: 6.0.2: the "Request Certificate" option could break TLS 1.2 negotiations.
- Bug Fix: GSSAPI: 5.3: SMTP/POP/IMAP/LDAP GSSAPI/Kerberos authentication could fail.
- Bug Fix: Airsync: 5.4.7: server could fail to read message IDs in requests.
- Bug Fix: CGPL: 5.4: non-printable characters might be converted to JSON incorectly.
- Bug Fix: KERNEL: 4.3: NTLM hashes might be calculated incorrectly.
- Bug Fix: TLS: 6.0: outgoung connections might fail to negotiate TLS parameters.

**6.0.8** 27-Dec-2013
`Valid Core License Keys`: issued after 01-Dec-2011.

- MAPI: the [MAPI Connector version 1.54.11.5](#) is included.
- Pronto: [Pronto! Version 6.0.8 and Pronto Lite 2.1](#) are included.
- FileDAV: reading (GET/HEAD) the Account root directory returns an empty file instead of the 404 error (CyberDuck client compatibility).
- TLS: ciphers offered for outgoing connections have been reordered.
- TLS: the CA chain sent with the server certificate is processed now.
- CLI: the GETDNRSETTINGS, SETDNRSETTINGS commands are implemented.
- Kernel: additional filtering of composed XML data is implemented.
- XIMSS: the contactFind "fuzzy" search is imporved, the returned data message attributes are extended.
- Triggers: sent IMs are pre-authorized using the postmaster Account.
- Bug Fix: AIRSYNC: 6.0.c2: "flagged" message flag was reported in pre-12.0 protocols, causing problems for old AirSync clients.
- Bug Fix: SMTP: 5.3c2: the SPF check procedure did not verify AAAA records of an MX if an A-record lookup was successful.
- Bug Fix: SMTP: 6.0c3: the reverse-connect procedure checked the highest MX relay only.
- Bug Fix: PBX: 5.2: the auto-attendant script would stop on unexpected INFO events.
- Bug Fix: WebMail: 6.0.7: the folder tree of a single INBOX might appear empty in Simplex and derived skins.
- Bug Fix: DAV: 6.0c1: the server could crash while searching the Directory.
- Bug Fix: CLUSTER: 6.0c1: synchronization did not work in Cluster (virtual) mailboxes.
- Bug Fix: MAILBOX: 6.0c1: when a Sliced-type mailbox was deleted, its size was calculated incorrectly.
- Bug Fix: MEDIAPROXY: 6.0c1: proxy for "udptl" media streams were not built.
- Bug Fix: TLS: 5.4c1: secure re-negotiation could fail.

**6.0.7** 10-Sep-2013
`Valid Core License Keys`: issued after 01-Dec-2011.

- Pronto: [Pronto! Version 6.0.7](#) is included.
- MediaServer: the "mixMOH" option is implemented.
- MediaServer: the "icePassword" data is randomized now.

MediaProxy: the "icePassword" data is randomized now.
- AIRSYNC: attachments in multipart/signed messages were not reported to clients.
- Bug Fix: KERNEL: 5.2: ISO-2022-KR encoding could work incorrectly.
- Bug Fix: WebMail: Mailboxes might be sorted incorrectly in Simplex and derived skins.
- Bug Fix: CLUSTER: when a new server was added, it did not correctly process already suspended domains.

**6.0.6** 16-Aug-2013
`Valid Core License Keys:` issued after 01-Dec-2011.

- MAPI: the MAPI Connector version 1.54.11.3 is included.
- Pronto: Pronto! Version 6.0.6 is included.
- WebMail: "style"-data filtering has been improved.
- CalDAV: Task-type mailbox creation and retrieval is supported now.
- CalDAV: mailbox deletion is implemented.
- CardDAV: mailbox deletion is implemented.
- HTTP: RFC6585 is implemented.
- CLI: the LISTSTOCKSKINFILES,READSTOCKSKINFILE,LISTSTOCKPBXFILES,READSTOCKPBXFILE commands are implemented.
- MAILBOX: the "delete folder only" operation is implemented.
- Bug Fix: SIP: 5.0.4: IPv6 URI processing could fail.
- Bug Fix: CLUSTER: 6.0.1: after a failover the new active controller could not immediately assign a new backup controller.
- Bug Fix: AIRSYNC: 6.0.2: voice mail attachments might be not reported to mobile devices.
- Bug Fix: CLI: 5.4: setting values with SETSTATELEMENT command did not work.
- Bug Fix: CALDAV: 6.0c1: support for old Apple CalDAV clients was broken.
- Bug Fix: KERNEL: 6.0c1: some Japanese characters were encoded incorrectly.

**6.0.5** 08-May-2013
`Valid Core License Keys:` issued after 01-Dec-2011.

- MAPI: the MAPI Connector version 1.54.11.1 is included.
- Pronto: Pronto! Version 6.0.5 is included.
- AIRSYNC: application/vnd.ms-sync.multipart responses are implemented.
- AIRSYNC: now "oldest items" sync limitations are applied to contents of calendaring items, not to the item timestamps.
- CLUSTER: simplified nonce processing for static clusters is implemented.
- CLUSTER: domain-level updates are delivered using size-limited chunks now.
- Kernel: the Mailer-Daemon messages use the sender domain name wherever possible.
- Bug Fix: Kernel: 6.0.2: RFC2231-encoded multi-part parameters were processed incorrectly.
- Bug Fix: XMPP: 6.0c2: sending packets to "XMPP components" could crash the server.
- Bug Fix: POP: 6.0.1: the POP Login Limit error was not reported in Cluster environments.
- Bug Fix: WebUser: 6.0c2: array-type Contact fields were updated incorrectly.
- Bug Fix: CLUSTER: 6.0c1: some static cluster operations worked incorrectly.
- Bug Fix: HTTP: 5.2: "chunked"-requests were proxied incorrectly.
- Bug Fix: PBXApp: 5.3: call termination during an "audio recording" operation could cause server crashing.

**6.0.4** 28-Mar-2013
`Valid Core License Keys:` issued after 01-Dec-2011.

- CLI: the GETLOGSETTINGS, UPDATELOGSETTINGS commands are implemented.
- CLI: the GETSERVERSETTINGS, UPDATESERVERSETTINGS commands are documented.
- ACCOUNT: OS-prohibited ASCII symbols in file names are encoded now.
- QUEUERULES: the Accept Reply action is implemented.
- Bug Fix: ACCOUNT: 6.0.2: account password restrictions were checked incorrectly.

Bug Fix: Skins: 6.0.3: language file processing was broken.
- Bug Fix: Kernel: 6.0.2: RFC2231-encoded parameters could be decoded twice.

**6.0.3** 26-Mar-2013
`Valid Core License Keys:` issued after 01-Dec-2011.

- Pronto: [Pronto! Version 6.0.3](#) is included.
- CalDAV: calendar delegation is implemented.
- SMTP: the Default IPv6 Source Address setting is implemented.
- RPOP: the Default IPv6 Source Address setting is implemented.
- XMPP: the Default IPv6 Source Address setting is implemented.
- HTTP: RFC5785 is implemented.
- AIRSYNC: html-embedded objects from "related" MIME parts are reported to clients now.
- WebDAV: the "principal" data processing is improved.
- WebDAV: the "expand-property" REPORTs are implemented.
- Bug Fix: XMPP: 6.0.2: if the peer closed connection during a sender validation procedure, the server could crash.
- Bug Fix: AirSync: 6.0c2: if an account contained "non-syncable" mailboxes, some syncable mailboxes disappeared from device view, too.
- Bug Fix: Mailbox: 6.0c1: when many items were deleted at once, some items might remain in the mailbox.

**6.0.2** 06-Mar-2013
`Valid Core License Keys:` issued after 01-Dec-2011.

- TLS: TLS v1.2 (RFC5246) is implemented.
- TLS: Maximum Fragment Length Negotiation extension (RFC4366) is implemented.
- CLI: the SETACCOUNTPASSWORD command checks the password size and complexity restrictions only if the CHECK keyword is used.
- MediaProxy: the ICE protocol is supported now.
- MediaProxy: STUN packet processing is implemented.
- MediaServer: the ICE protocol is supported now.
- MediaServer: STUN packet processing is implemented.
- PBXApp: the new MediaOption function replaces the MediaChannelOption function.
- PBXApp: the SignalOption function is implemented.
- PBXApp: the ProvisionCallEx function is renamed back into ProvisionCall.
- PBXApp: the SetForeignCredentials, DisconnectEx, SetReferMode, SetTransferReportMode, SetBridgeBreakMode, SetBridgedProvision procedures are deprecated.
- PBXApp: media encryption is enabled by default.
- NETWORK: the predefined LAN IP addresses now include only the networks in the 10/8, 172.16/12, 192.168/16 ranges.
- XMPP: multiple validated domains per incoming connection are supported now.
- Kernel: HTTP, SIP "Server" header fields are customizable now.
- Kernel: RFC2231 processing has been improved.
- CLUSTER: the Balancer Weight option is implemented.
- Foundation: raw Unicode strings in certificate names are supported now.
- WebAdmin: the startup options are displayed on the Settings->General->Info page.
- AIRSYNC: message bodies are retrieved in the original HTML format for 12.0 and newer clients.
- AIRSYNC: support for Windows Phone 8 has been implemented.
- XIMSS: callConnected, callDisconnected, callProvisioned, callIncoming messages now contain the callId attribute.
- Bug Fix: AIRSYNC: 6.0c2: modification waiting operations (PING, SYNC) were interrupted by unrelated SYNC operations.
- Bug Fix: AIRSYNC: 6.0c2: a mailbox tree with new mailboxes in foreign accounts could be synchronized incorrectly.

- Bug Fix: QUEUERULES: 6.0.1: messages created with "Redirected" Rule did not get the "trusted source" flag and could fail.
- Bug Fix: MediaServer: 6.0c1: the Secure RTP AES_CM_128_HMAC_SHA1_32 suite incorrectly processed SRTCP packets.
- Bug Fix: DIRECTORY: 6.0c1: the Remote Unit could open LDAP connections using the intra-cluster IP address.
- Bug Fix: PIPE: 6.0.1: return-path processing was broken.
- Bug Fix: TLS: 6.0c2: additional checks were incorrectly applied to SSLv3 connections.

**6.0.1** 30-Dec-2012
`Valid Core License Keys:` issued after 01-Dec-2011.

- Pronto: [Pronto! Version 6.0.1](#) is included.
- Pronto Mobile: Pronto! for Android 1.8 is released.
- Dialogs: call expiration is set only when both sides have indicated support for a session expiration timer.
- PBXApp: the MuteMixer procedure is extended to implement "whisper" services.
- IMAP: the initial prompt contains the incoming domain name now.
- SMTP: the input prompts contain the incoming domain name now.
- SMTP: the HELO name now contains the secondary domain name for domains with assigned IP addresses.
- POP: the POP Login Limit Account setting is implemented.
- Kernel: E-mails submitted internally always contain the submitter domain name now.
- XIMSS: the "cookie" listDirectory attribute is documented.
- PBXApp: the Call-Info header in provision responses for StartBridgedCall() is copied to the ingress response.
- FTP: the Use Fixed Active Port option is implemented.
- Statistics: the global network elements (total TCP and UDP data elements) are implemented.
- CG/PL: the Convert function now supports "nonce" operations.
- Bug Fix: ACAP: 6.0c2: the initial greeting was not sent till the first command was received.

**6.0** 14-Dec-2012
`Valid Core License Keys:` issued after 01-Dec-2011.

- Pronto: [Pronto! Version 6.0](#) is included.
- Pronto Mobile: Pronto! for Android 1.7 is released.
- HTTP: the "Access" setting is implemented for sub-protocols (AirSync, DAV, XIMSS, WebUser).
- DOMAINS: the former SMTP Relay setting is re-used as the Use IP for Outgoing Connections setting.
- DOMAINS: incoming and outgoing call statistics elements have been added.
- Foundation: String object representations can include \u'xxxx' symbol combinations (hexadecimal Unicode characters).
- Foundation: JSON strings now represent control characters as \u00nn symbol combinations.
- MediaProxy: several create/destroy operations can be executed concurrently now.
- MediaProxy: the "media idle" time is calculated separately for each direction, RTCP packets do not reset the idle time counter now.
- Dialogs: when a dialog object detects an "idle timeout" condition, it sends BYE requests to both call sides.
- PBXApp: the PlayTone procedure is implemented.
- PBXApp: the PlayFileInLoop and PlayInLoop procedures are deprecated.
- Kernel: the --lockLockFile command line option is implemented.
- Bug Fix: XMPP: 6.0c3: WebAdmin displayed the "illegal parameter value" error if the Use Domain IP Address option was not selected.
- Bug Fix: Groupware: 6.0c2: some "Real Names" parameters in "Contact Groups" could not be read.

**6.0c3** 02-Dec-2012
`Valid Core License Keys:` issued after 01-Dec-2011.

- MAPI: the new "SYNC"-type version has been implemented (beta).

Pronto: [Pronto! Version 6.0c3](#) is included.

- Foundation: Number objects can be represented as decimal, hexadecimal, octal, or binary numbers.
- HTTP: the Compress If Larger option (response gzip-compression) is implemented.
- Kernel: source IP address selection for outgoing connections supports IPv6 targets now.
- Skins: the javascript and css files are deflated on load (comments and excessive space symbols removed).
- TLS: TLS v1.1 (RFC4346) is implemented.
- TLS: the TLS CRC Ciphers option has been changed into CRC Ciphers for old TLS, CRC Ciphers are always supported for TLS v1.1 and higher.
- XIMSS: the spellerCheck command now supports the "mode" attribute.
- WSSP: the ServerVersion() function is implemented.
- CG/PL: the SystemInfo function is implemented.
- CLI: the GETSYSTEMINFO command is implemented.
- Media: RFC5761 (Multiplexed RTCP) is implemented.
- Media: RFC3605 (RTCP attribute) is implemented.
- XIMSS: the sdpText element can be used instead of the sdp XML element.
- AIRSYNC: initial support for Outlook 2013 has been implemented.
- AIRSYNC: Sync command for protocol version 14.0 is supported.
- AIRSYNC: the Search command for the message store is implemented.
- AIRSYNC: the "Flagged" message flag is supported now.
- HELPERS: if the server is started with the --dropRoot flag, the Helper applications are not started till the root privilege is dropped.
- DOMAINS: the SMTP Relay setting is removed.
- SIP: if a request has been authenticated, "unknown account"-type errors do not result in "banning" of the client IP address.
- Bug Fix: MAILBOX: 6.0c1: appending messages to Sliced Mailboxes could crash the server.

**6.0c2** 07-Oct-2012

`Valid Core License Keys:` issued after 01-Oct-2011.

- Pronto: [Pronto! Version 6.0c2](#) is included.
- CLUSTER: the "Admin Connections" setting has been removed.
- CLUSTER: the Dynamic Cluster administrative processing has been reimplemented.
- CLUSTER: support for Software Load Balancers is implemented.
- CLUSTER: Linux Software Load Balancer Helper (based on "IPVS" kernel balancer) is implemented.
- CLUSTER: static cluster processing has been redesigned.
- NETWORK: port allocation mechanism has been redesigned.
- XIMSS: the SDP data element now can contain `crypto` sub-elements.
- XIMSS: the "messageCopy" operation now supports the "decrypt" attribute.
- WebUser: the mailbox sorting mechanism has been improved.
- WebUser: the "Redirect/SRedirect" mechanism now uses "nonces" to prevent "HTTP redirection server" abuses.
- WebUser: the Edit Event/Edit Task operations now preserve the original item UID.
- WebAdmin: the Proxy configuration settings have been moved to the "NAT" page.
- GROUPWARE: multiple VALARM elements per VEVENT object are supported now.
- MEDIA: the rtp/avpf and rtp/savpf protocols are recognized now.
- CG/PL: the "var" definitions allow initial value assignments.
- CG/PL: the "for-loop" can now use a "var" declaration as its "initializer operator".
- CG/PL: all loop operator sequences between the exitif operators, and the exitif operators themselves now belong to one single "operator block".
- CG/PL: DNR operations have been added to the Convert function.
- CLI: the LISTACCOUNTSESSIONS command is implemented.
- XMPP: the full dialback receiver functionality is implemented.
- XMPP: the Source IP Address settings are implemented.
- ACCOUNT: when Account service files (.settings, .info) are deleted, the shadow ".temp" copies are deleted, too.
- TLS: the CRC Ciphers setting is implemented.

- Bug fix: WebUser: real names in To/Cc addresses might be encoded incorrectly for Reply and Reply All operations.

**6.0c1** 20-Jul-2012

`Valid Core License Keys:` issued after 01-Jul-2011.

- Pronto: Pronto! v.3 has been discontinued.
- Kernel: the PacketCable 2.0 Lawful Intercept protocol is implemented.
- Kernel: the iso-2022-jp-1, iso-2022-jp-2 (RFC1554), iso-2022-jp-3 encodings are supported now.
- Kernel: the GB18030 encoding is supported now.
- Foundation: now C-style (/*...*/) comments are supported in object textual presentations.
- ACCOUNT: the Minimal Password Length and Password Complexity settings have been implemented.
- ACCOUNT: default Class of Service settings are implemented.
- MAILBOX: the Sliced Mailbox format is implemented.
- MAILBOX: the Erase Deleted Messages Account Setting is implemented.
- EXTMAILBOX: the AirSync "provisioning required" responses are supported now.
- QUEUE: message queue management for LIST, LOCAL, PIPE, SMTP modules has been improved.
- SMTP: recovery after a failed STARTTLS operation has been improved.
- SIGNAL: when an IM request is directed to a resource, and that resource has been disconnected, the IM is delivered to all connected resources.
- MediaServer: SRTP support (encrypted media, RFC3711, RFC4568) is implemented.
- SIP: the TCPPing workaround is implemented.
- XIMSS: the SMIMEGet and SMIMESet operations are implemented; the Private Key and Certificate data can be exported now.
- XMPP: failed message delivery reports are relayed to external senders.
- SMPP: all types of incoming message UDH are supported now.
- SMPP: multi-part messages received in a Cluster are reassembled on the Controller now.
- PBXApp: the TimerB, P-Billing-Id, Relay, Via, No100rel StartCall() parameters are documented.
- PBXApp: the ProvisionCallEx function is implemented, the SetProvisionCode procedure is deprecated.
- PBXApp: the "allowedAudioCodecs", "allowedVideoCodecs", "encryptMedia" call parameters are implemented.
- PBX: the "gatewaycaller" and "gatewayincoming" applications remove video media and non-PCMU audio codecs from SDP sent to PSTN gateways.
- WebUser: E-mail headers are displayed for non-PUBLISH Event and Task messages.
- CLI: the LISTCLICOMMANDS, NOOP, ECHO, GETVERSION, GETCURRENTTIME, SETLOGALL, DUMPALLOBJECTS, TESTLOOP commands are documented.
- CardDAV: the GET method is implemented.
- CLUSTER: the Backend LDAP port setting is implemented.

# Summary

**Foundation**

- C-style (/*...*/) comments are supported in object textual presentations.
- Number objects can be represented as decimal, hexadecimail, octal, or binary numbers.
- String object representations can include \u'xxxx' symbol combinations (hexadecimal Unicode characters).
- JSON strings now represent control characters as \u00nn symbol combinations.
- raw Unicode strings in certificate names are supported now.

**Kernel**

- the PacketCable 2.0 Lawful Intercept protocol is implemented.
- the iso-2022-jp-1, iso-2022-jp-2 (RFC1554), iso-2022-jp-3 encodings are supported now.
- the GB18030 encoding is supported now.

- source IP address selection for outgoing connections supports IPv6 targets now.
- the --lockLockFile command line option is implemented.
- HTTP, SIP "Server" header fields are customizable now.
- E-mails submitted internally always contain the submitter domain name now.
- RFC2231 processing has been improved.
- additional filtering of composed XML data is implemented.

## Domains

- the SMTP Relay setting is re-used as the Use IP for Outgoing Connections setting.
- incoming and outgoing call statistics elements have been added.

## Accounts

- the Minimal Password Length and Password Complexity settings have been implemented.
- default Class of Service settings are implemented.
- when Account service files (.settings, .info) are deleted, the shadow ".temp" copies are deleted, too.
- OS-prohibited ASCII symbols in file names are encoded now.

## Mailbox

- the Sliced Mailbox format is implemented.
- the Erase Deleted Messages Account Setting is implemented.
- the "delete folder only" operation is implemented.

## External Mailbox

- the AirSync "provisioning required" responses are supported now.

## NETWORK

- port allocation mechanism has been redesigned.
- the predefined LAN IP addresses now include only the network in the 10/8, 172.16/12, 192.16/16 ranges.

## CLUSTER

- the Backend LDAP port setting is implemented.
- the "Admin Connections" setting has been removed.
- the Dynamic Cluster administrative processing has been reimplemented.
- support for Software Load Balancers is implemented.
- Linux Software Load Balancer Helper (based on "IPVS" kernel balancer) is implemented.
- static cluster processing has been redesigned.

## QUEUE

- message queue management for LIST, LOCAL, PIPE, SMTP modules has been improved.

## QUEUERULES

- the Accept Reply action is implemented.

## SIGNAL

- when an IM request is directed to a resource, and that resource has been disconnected, the IM is delivered to all connected resources.

## GROUPWARE

- multiple VALARM elements per VEVENT object are supported now.

## Pronto

- Pronto! v.3 has been discontinued.
- [Pronto! Version 6.0 and Pronto Lite 2.1](#) are included.
- Pronto Mobile: Pronto! for Android 1.8 is released.

## MAPI

- new server-side libraries have been added.

## AIRSYNC

- initial support for Outlook 2013 has been implemented.
- Sync command for protocol version 14.0 is supported.
- the Search command for the message store is implemented.
- the "Flagged" message flag is supported now.
- message bodies are retrieved in the original HTML format for 12.0 and newer clients.
- support for Windows Phone 8 has been implemented.
- html-embedded objects from "related" MIME parts are reported to clients now.
- application/vnd.ms-sync.multipart responses are implemented.

## CalDAV

- calendar delegation is implemented.
- Task-type mailbox creation and retrieval is supported now.
- mailbox deletion is implemented.

## CardDAV

- the GET method is implemented.
- mailbox deletion is implemented.

## FileDAV

- reading (GET/HEAD) the Account root directory returns an empty file instead of the 404 error (CyberDuck client compatibility).

## PBXApp

- the TimerB, P-Billing-Id, Relay, Via, No100rel StartCall() parameters are documented.
- the ProvisionCall function is extended.
- the PlayTone procedure is implemented.
- the PlayFileInLoop and PlayInLoop procedures are deprecated.
- the "allowedAudioCodecs", "allowedVideoCodecs", "encryptMedia" call parameters are implemented.
- the MuteMixer procedure is extended to implement "whisper" services.
- the Call-Info header in provision responses for StartBridgedCall() is copied to the ingress response.
- the SetForeignCredentials, SetProvisionCode, DisconnectEx, SetReferMode, SetTransferReportMode, SetBridgeBreakMode, SetBridgedProvision procedures are deprecated.
- the new MediaOption function replaces the MediaChannelOption function.
- the SignalOption function is implemented.
- media encryption is enabled by default.

**MediaProxy**

- several create/destroy operations can be executed concurrently now.
- the "media idle" time is calculated separately for each direction, RTCP packets do not reset the idle time counter now.
- the ICE attributes are removed from the modified SDP data.
- the ICE protocol is supported now.
- STUN packet processing is implemented.

**MediaServer**

- SRTP support (encrypted media, RFC3711, RFC4568) is implemented.
- the rtp/avpf and rtp/savpf protocols are recognized now.
- RFC5761 (Multiplexed RTCP) is implemented.
- RFC3605 (RTCP attribute) is implemented.
- the ICE protocol is supported now.
- STUN packet processing is implemented.
- the "mixMOH" option is implemented.

**Dialogs**

- when a dialog object detects an "idle timeout" condition, it sends BYE requests to both call sides.
- call expiration is set only when both sides have indicated support for a session expiration timer.

**HTTP**

- the Compress If Larger option (response gzip-compression) is implemented.
- the "Access" setting is implemented for sub-protocols (AirSync, DAV, XIMSS, WebUser).
- RFC5785 is implemented.
- RFC6585 is implemented.

**WebDAV**

- the "principal" data processing is improved.
- the "expand-property" REPORTs are implemented.

**SIP**

- the TCPPing workaround is implemented.
- if a request has been authenticated, "unknown account"-type errors do not result in "banning" of the client IP address.

**SMTP**

- recovery after a failed STARTTLS operation has been improved.
- the input prompts contain the incoming domain name now.
- the HELO name now contains the secondary domain name for domains with assigned IP addresses.
- the Default IPv6 Source Address setting is implemented.

**IMAP**

- the initial prompt contains the incoming domain name now.

**POP**

the POP Login Limit Account setting is implemented.

## RPOP

- the Default IPv6 Source Address setting is implemented.

## XMPP

- failed message delivery reports are relayed to external senders.
- the full dialback receiver functionality is implemented.
- the Source IP Address settings are implemented.
- multiple validated domains per incoming connection are supported now.
- the Default IPv6 Source Address setting is implemented.

## SMPP

- all types of incoming message UDH are supported now.
- multi-part messages received in a Cluster are reassembled on the Controller now.

## FTP

- the Use Fixed Active Port option is implemented.

## WebAdmin

- the Proxy configuration settings have been moved to the "NAT" page.
- the startup options are displayed on the Settings->General->Info page.

## WebUser

- E-mail headers are displayed for non-PUBLISH Event and Task messages.
- the mailbox sorting mechanism has been improved.
- the "Redirect/SRedirect" mechanism now uses "nonces" to prevent "HTTP redirection server" abuses.
- the Edit Event/Edit Task operations now preserve the original item UID.
- "style"-data filtering has been improved.

## XIMSS

- the SMIMEGet and SMIMESet operations are implemented; the Private Key and Certificate data can be exported now.
- the SDP data element now can contain `crypto` sub-elements.
- the "messageCopy" operation now supports the "decrypt" attribute.
- the spellerCheck command now supports the "mode" attribute.
- the sdpText element can be used instead of the sdp XML element.
- the "cookie" listDirectory attribute is documented.
- callConnected, callDisconnected, callProvisioned, callIncoming messages now contain the callId attribute.
- the contactFind "fuzzy" search is imporved, the returned data message attributes are extended.

## CLI

- the LISTCLICOMMANDS, NOOP, ECHO, GETVERSION, GETCURRENTTIME, SETLOGALL, DUMPALLOBJECTS, TESTLOOP commands are documented.
- the LISTACCOUNTSESSIONS command is implemented.
- the GETSYSTEMINFO command is implemented.
- the SETACCOUNTPASSWORD command checks the password size and complexity restrictions only if the

CHECK keyword is used.

- the GETLOGSETTINGS, UPDATELOGSETTINGS commands are implemented.
- the GETSERVERSETTINGS, UPDATESERVERSETTINGS commands are documented.
- the LISTSTOCKSKINFILES,READSTOCKSKINFILE,LISTSTOCKPBXFILES,READSTOCKPBXFILE commands are implemented.
- the GETDNRSETTINGS, SETDNRSETTINGS commands are implemented.

## Skins

- the javascript and css files are deflated on load (comments and excessive space symbols removed).

## WSSP

- the ServerVersion() function is implemented.

## CG/PL

- the "var" definitions allow initial value assignments.
- the "for-loop" can now use a "var" declaration as its "initializer operator".
- all loop operator sequences between the exitif operators, and the exitif operators themselves now belong to one single "operator block".
- DNR operations have been added to the Convert function.
- the SystemInfo function is implemented.
- the Convert function now supports "nonce" operations.

## TLS

- the CRC Ciphers setting is implemented.
- TLS v1.1 (RFC4346) is implemented.
- the CRC Ciphers option has been changed into CRC Ciphers for old TLS, CRC Ciphers are always supported for TLS v1.1 and higher.
- TLS v1.2 (RFC5246) is implemented.
- Maximum Fragment Length Negotiation extension (RFC4366) is implemented.
- ciphers offered for outgoing connections have been reodered.
- the CA chain sent with the server certificate is processed now.

## STATISTICS

- the global network elements (total TCP and UDP data elements) are implemented.
- sent Trigger IMs are pre-authorized using the postmaster Account.

## HELPERS

- if the server is started with the --dropRoot flag, the Helper applications are not started till the root privilege is dropped.

## PBX

- the "gatewaycaller" and "gatewayincoming" applications remove video media and non-PCMU audio codecs from SDP sent to PSTN gateways.

---

| Introduction | Features | History | How To | Help Me |
|---|---|---|---|---|

# Version 5.4 Revision History

- **6.2**
- **6.1**
- **6.0**
- **5.4: Summary**
- **5.3**
- **5.2**
- **5.1**
- **5.0**
- **4.x**
- **3.x**
- **2.x**
- **1.x**

- **Pronto!**
- **MAPI Connector**

RSS

**Note:** when upgrading from 5.3 and earlier versions, AirSync accounts on Apple iPhones should be removed and re-configured.

**5.4.11** 18-Oct-2013
`Valid Core License Keys:` issued after 01-Jun-2010.

- MAPI: the MAPI Connector version 1.54.11.4 is included.
- Pronto: Pronto! Version 4.11 is included.
- Signal: new response session-expiration values are processed now.
- Bug Fix: WebAdmin: 5.3: outdated "Enabled Mode" services were displayed incorrectly.
- Bug Fix: CLI: 5.2: setting values with SETSTATELEMENT command did not work.
- Bug Fix: WebMail: 5.1: Mailboxes might be sorted incorrectly in Simplex and derived skins.
- Bug Fix: CLUSTER: 5.2: SIP INVITE-replaces requests could loop in a SIP Farm.
- Bug Fix: MAILBOX: 5.3: message attribute modification could be incorrectly processed in "sorted" mailbox views.
- Bug Fix: Kernel: 4.0: Digest-MD5 calculations were incorrect if the "realm" string contained semicolon symbols.

**5.4.10** 28-Mar-2013
`Valid Core License Keys:` issued after 01-Jun-2010.

- Pronto: Pronto! Version 4.10 is included.
- Bug Fix: MAILBOX: 5.3: failures to store a message attribute item could cause server crashes.

Bug Fix: CLUSTER: 5.3: Named Tasks running on a disconnected frontend were not removed from the Controller tables.

- Bug Fix: TLS: 5.2: incorrectly composed TLS handshake records could crash the server.
- Bug Fix: HTTP: 5.1: rejected inter-cluster connection could result in memory leaks.
- Bug Fix: AIRSYNC: 5.3: requests with some base64-encoded URL parameters could fail.
- Bug Fix: DOMAINS: 5.3: account auto-creation in Directory Domains could fail.

### 5.4.9 30-Dec-2012
`Valid Core License Keys`: issued after 01-Jun-2010.

- Pronto: Pronto! Version 4.9 is included.
- Bug Fix: DOMAIN: 5.0: the RESETSTATISTIC operation did not reset "sent email" data.
- Bug Fix: INTERCEPT: 5.3: after a "monitored" Account authenticated via SMTP, other Accounts re-using the same stream object could be monitored, too.
- Bug Fix: POP: 5.3: XTND XMIT command could result in an incorrect object release operation.

### 5.4.8 06-Oct-2012
`Valid Core License Keys`: issued after 01-Jun-2010.

- MAPI: the MAPI Connector version 1.54.8.1 is included.
- Foundation: the cp932 charset is supported now.
- SIP: the 503-type response is processed as a regular one if there is no more addresses to try; this is done to preserve the response Reason: field.
- Bug Fix: WebUser: 5.2c1: real names in To/Cc addresses might be encoded incorrectly for Reply and Reply All operations.
- Bug Fix: WebUser: 5.1c1: converting pages to HTML could work incorrectly for some two-byte charsets.
- Bug Fix: CG/PL: 5.3c1 moving XML nodes between parent nodes could crash the server.
- Bug Fix: CLI: 5.3c2: the SetLogAll CLI command incorrectly processed its parameters.

### 5.4.7 24-Aug-2012
`Valid Core License Keys`: issued after 01-Jun-2010.

- Pronto: Pronto! Version 4.7 is included.
- The Pronto! HTML5 version 1.4 is released.
- Pronto Mobile: Pronto! for Android 1.4 is released.
- SMTP: outgoing STARTTLS connections now use both the TLSv1 and SSLv3 protocols.
- SIGNAL: the "ActiveHold" workaround now works when a MOH channel is not created for that call leg.
- Bug Fix: STREAM: 5.4c1: "concurrent connections from the same IP" calculations could result in a server crash.
- Bug Fix: XIMSS: 5.4c2: composing a large encrypted message could fail.
- Bug Fix: SIP: 5.4.5: URI "userpart parameters" were composed incorrectly.
- Bug Fix: AirSync: 5.2c2: Delete and Move operations for recently modified messages could fail.

### 5.4.6 21-Jun-2012
`Valid Core License Keys`: issued after 01-Jun-2010.

- Pronto: Pronto! Version 4.6 is included.
- Pronto Mobile: Pronto! for iPhone 1.3 has been released (via Apple Store).
- Pronto Mobile: Pronto! for Android 1.3 is released.
- Foundation: a workaround for Solaris 11 realloc() bug.
- MAPI: the MAPI Connector version 1.54.6.2 is included.
- Bug Fix: INTERCEPT: 5.3: after a "monitored" Account logged in via a stream-based protocol, other Accounts re-using the same stream object could be monitored, too.
- Bug Fix: XIMSS: 5.1: the messageSubmit operation ignored a timeShift attribute of Date-type fields if a localTime attribute was present, too.
- Bug Fix: XIMSS: 5.1: the messageSubmit operation did not generate the MIME-VERSION header if it was not

specified explicitly.

**5.4.5** 26-Apr-2012
`Valid Core License Keys:` issued after 01-Jun-2010.

- The Pronto! HTML5 version is released.
- Pronto Mobile: Pronto! for iPhone 1.2 has been released (via Apple Store).
- Pronto Mobile: Pronto! for Android 1.2 is released.
- Pronto: Pronto! Version 4.5 is included.
- MAPI: the MAPI Connector version 1.54.5.5 is included.
- SIP: URI parsing/composing is improved.
- CalDAV: Mozilla Lightning compatibility is improved.
- AIRSYNC: S/MIME compatibility with some Android clients is improved.
- AIRSYNC: some Unicode symbols are filtered out, as Apple devices crash when receiving those symbols.
- SIGNAL: support for "collapsable proxies" is added to B2BUA calls made in the "connected" state.
- FTP: the SITE NAMEFMT and SITE LISTFMT commands are implemented now (OS/400 client support).
- Bug Fix: WebDAV: 5.4c1: the "If" HTTP header could be processed incorrectly.
- Bug Fix: CLUSTER: 5.1: some ADMIN-type inter-cluster commands in a Static Cluster could be sent without an Account name.
- Bug Fix: XIMSS: 5.3.1: HTTP-binding "CRAMMD5" login could fail in a Static Cluster.
- Bug Fix: SMTP: 5.1: after an incoming SMTP session was authenticated using an Account hosted on a different Static Cluster server, some new SMTP sessions could fail trying to submit received messages.
- Bug Fix: Kernel: 5.0: when composing address-type E-mail header fields, converting to the ISO-2022 charsets could work incorrectly.
- Bug Fix: PBXLEG: 5.3c3: when an INVITE-replaces request is composed the SDP session-level direction attribute was not reset.
- Bug Fix: EXTERNAL: 5.4c2: AirSync mailbox tree synchronization could fail.
- Bug Fix: ACCOUNT: 2.0: under very heavy load retrieving an Account Info set could crash the server.
- Bug Fix: XIMSS: 5.3: signed/encrypted message copies could be truncated when they were stored in the "Sent" Mailbox.
- Bug Fix: EXTERNAL MAILBOX: 5.4.1: external IDs assigned to newly created mailboxes could be assigned to the INBOX Mailbox instead.

**5.4.4** 02-Feb-2012
`Valid Core License Keys:` issued after 01-Jun-2010.

- Pronto: Pronto! Version 4.4 is included.
- Pronto Mobile: Pronto! for iPhone has been released (via Apple Store).
- MAPI: the MAPI Connector version 1.54.4.3 is included.
- Kernel: 32-bit Unicode symbols are supported now.
- MEDIAPROXY: the UDP Port Restricted option now applies to non-NAT clients only.
- NETWORK: the TCP Listener "connection aborted"-type errors are logged, but ignored now.
- SIP: incorrectly composed response packets do not result in IP address blocking now.
- AIRSYNC: the ItemOperation Fetch operation for retrieving a message MIME body is implemented.
- HTTP: the RootDAV and UserDAV Log-Level settings are implemented.
- Bug Fix: MAILBOX: 5.4c1: moving messages between mailboxes in the same Account did not switch off mailbox store size limit check.
- Bug Fix: AIRSYNC: 5.4c1: fetch parameters could be processed incorrectly.
- Bug Fix: AIRSYNC: 5.4c1: AirSync "provision" server response elements were provided in an incorrect order.
- Bug Fix: WebUser: 5.4.3: an attempt make a duplicate using the "Take Certificate" operation could remove the entire vCard from a Mailbox-type Address Book.
- Bug Fix: WebUser: 5.4c1: non-ASCII mailbox names could be sorted incorrectly.
- Bug Fix: RPOP: 5.4c2: RPOP records with names containing spaces did not work in Dynamic Clusters.
- Bug Fix: GROUPWARE: 4.2: a date-only iCalendar field value could be decreased by 1 day when the item is

being stored.

- Bug Fix: ACCOUNT: only the first 64Kb portion of iCalendar data could be read via the "pubcal/" virtual files.
- Bug Fix: SIP: 5.3.0: when a Proxy is built for an outgoing request and a negative response is received, the request SDP was not restored, causing problems for PBX-originated calls.

## 5.4.3 19-Dec-2011
`Valid Core License Keys:` issued after 01-Jun-2010.

- Mobile: Pronto! for Android 1.1 is included.
- Pronto: Pronto! Version 4.3 is included.
- MAPI: the MAPI Connector version 1.54.3.1 is included.
- WSSP: a quoted string can be used as an expression.
- ACCOUNT: the CHRONOS Tasks are implemented.
- CLI: the UPDATESCHEDULEDTASK, GETIPSTATE commands are implemented.
- CG/PL: optional parameters for the built-in functions are implemented, the FindSubstringEx function is removed.
- LOG: the Source IP Address setting is implemented.
- DNR: the Source IP Address setting is implemented.
- PBXLEG: the Record function now records audio data from all media legs.
- PBXLEG: the SendCallNotify function is implemented.
- WebUser: the Take Certificate operation does not store duplicates now.
- WebUser: the S/MIME signing operation accepts contacts with multiple X.509 certificates.
- NETWORK: the Relay for clients behind the same NAT option is implemented.
- Bug Fix: AIRSYNC: 5.4c1: the protocol v.12 "Sync" options could be processed incorrectly.
- Bug Fix: AIRSYNC: 5.4c1: if a message body was not truncated, the actual message size and the reported estimated size could be different.
- Bug Fix: ExtMailbox: 5.4c1: non-cluster IMAP APPENDs contained incorrect byte counts if the host OS used 1-byte EOLs.
- Bug Fix: Account: 5.4c1: updating Mailbox Message attributes could cause server crashes.

## 5.4.2 23-Oct-2011
`Valid Core License Keys:` issued after 01-Jun-2010.

- Pronto: Pronto! Version 4.2 is included.
- MAPI: the MAPI Connector version 1.54.0.6 is included.
- Media: the GIPS media engine License is now included into the Unified License Set.
- XIMSS: the fileStore operation has been extended.
- CG/PL: the FindSubstringEx, IsWhiteSpaces functions are implemented.
- AIRSYNC: devices using the "validate" client-id (such as Android clients) are auto-provisioned now.
- Bug Fix: HTTP: 5.1: session-less requests for the "sys" realm did not load Cluster-wide applications.
- Bug Fix: SMTP: 2.0: host name limit extended from 64 to 256 bytes.
- Bug Fix: WebUser: the "safe" redirection to external links might not work in Internet Explorer 8.
- Bug Fix: Kernel: 5.1: the server might crash parsing some malformed binary XML data (i.e. AirSync requests).
- Bug Fix: CG/PL: 5.1: the OBJECTTOJSON function produced non-standard "\e" symbols.
- Bug Fix: CG/PL: 5.2: the DIRECTORYSEARCH function could cause server crashes.
- Bug Fix: PBXApp: 5.0: the REMOTEURI and LOCALURI functions did not escape URIs.
- Bug Fix: PBXApp: 5.2: the value for Privacy header was not delivered to PBX applications.
- Bug Fix: Account: 5.3.3: new File Storage folder access rights were not copied from the outer folder if the creator was not the Account owner.
- Bug Fix: AIRSYNC: 5.4c1: Contacts created using AirSync client were stored with the "$Hidden" message flag set.
- Bug Fix: CHRONOS: 5.4c1: Account settings might be reset on a failure to auto-archive old messages.
- Bug Fix: XIMSS: 5.2: XIMSS clients could generate messages with multiple References and In-Reply-To header fields.

**5.4.1** 01-Aug-2011

`Valid Core License Keys:` issued after 01-Jun-2010.

- Pronto: Pronto! Version 4.1 is included.
- Foundation: now timestamps in remote past (like year 1902) do not cause certificate parsing failures.
- Kernel: now bang(uucp) E-mail addresses (such as host!user) are not supported by default, and the "!" symbol is interpreted as a regular one.
- XIMSS: file uploading using the HTTP PUT method is implemented.
- DOMAIN: alias/forwarder/telnum files with very long (over 32K) lines are supported now.
- ACCOUNT: improved RSIP and RPOP record processing.
- Bug Fix: ACCOUNT: 4.0: non-ASCII passwords could be encoded incorrectly if the BlowFish ("UB-crpt") encoding method was used.
- Bug Fix: LOCAL: 5.4.c1: when sending a message to "All" forwarders, a Return-Path field was added to the distributed message.
- Bug Fix: HTTPU: 5.4c1: OPTION requests for "in-session" resources could cause server crashes.
- Bug Fix: WSSP: 5.4.0: the CALENDARDATEMENU elements were composed incorrectly.
- Bug Fix: Calendar: 5.4.0: the server might crash while processing some recurrent events with exceptions.
- Bug Fix: AirSync: 5.4.0: the description of events created or updated with some mobile devices could be stored as an empty string.
- Bug Fix: IMAP: 5.2: fetching partial message body data when a message contained very long lines could fail.
- Bug Fix: ACCOUNT: 5.3: retrieving Account "special ACLs" when the Account settings were being modified could crash the server.

**5.4.0** 05-Jun-2011

`Valid Core License Keys:` issued after 01-Jun-2010.

- Pronto: Pronto! Version 4.0 is included.
- Pronto Mobile: Pronto! for Android 1.0b11 is included.
- MAPI: the MAPI Connector version 1.54.0.3 is included.
- WebUser: the mailbox menu elements now contain UTF-8 strings (instead of encoded UTF-7 strings).
- CALENDAR: if a VEVENT lacks the DTSTART property, it is added automatically.
- CLUSTER: the relayed requests to Cluster-wide Local Units now search that Unit only.
- NETWORK: all operations that supported DNS A-type records now support the AAAA-type records, too.
- XMPP: Secure (SSL/TLS) server-server communication is implemented.
- XIMSS: the mode attribute for the signalBind operation is supported now.
- XIMSS: the presenceSet operation now supports custom status messages.
- XIMSS: the Session Time Limit is applied to the TCP-based sessions, too.
- XIMSS: the setTimeout command is implemented, the readStatus command returns the session timeout information.
- XIMSS: the signalBind operation "deviceName" attribute has been renamed into "clientID" (the old attribute is accepted, too).
- Kernel: the --DelayOnStart startup option is implemented.
- Kernel: the --ExcludeLocalIP startup option is implemented.
- QUEUE: the pre-fetched subject size limit is increased to 2000 bytes.
- AIRSYNC: when a PRIVATE item is created or modified, the Hidden flag is added to the item message.
- Bug Fix: CalDAV: 5.4c2: ALARM elements were not sent to clients.
- Bug Fix: CalDAV: 5.4c1: when a POST'ed Event reply is sent, the E-mail From and To header fields were set incorrectly.
- Bug Fix: CALENDAR: 5.4c1: a REPLY operation applied to a recurrence exception could crash the Server.
- Bug Fix: CALENDAR: 5.4c1: recurrence exceptions in all-day events could be processed incorrectly.
- Bug Fix: CALENDAR: 5.4c1: x-item-id data could be created incorrectly.
- Bug Fix: STUN: 5.4c2: when an alternative listener was not specified, the server could crash.
- Bug Fix: MAILBOX: 5.4c2: some mailbox list operations could not find mailbox aliases.
- Bug Fix: WSSP: 5.4c1: MAILBOXMENU elements incorrectly processed non-ASCII selections.

- Bug Fix: DOMAIN: 5.3c1: Fast storage above the Domain directory was not renamed or removed on Domain rename/remove operations.

**5.4c2** 08-Mar-2011

`Valid Core License Keys:` issued after 01-Feb-2010.

- Pronto: [Pronto Version 4.0c2](#) is included.
- Signal: Access right checks for Event package subscriptions have been improved (see the Signal section of the manual)
- Signal: Offline Storage for Instant Messages is implemented.
- Signal: an Account Setting controlling Instant Messages delivery to SIP devices is implemented.
- Signal: an Account Preference controlling non-Roster senders of Instant Messages is implemented.
- XMPP: the XEP-114 extension is implemented (support for IM gateways).
- XMPP: the XEP-0203 (Delayed Delivery) and XEP-0160 (Offline Messages) are implemented.
- XMPP: the "Supplementary Discovery Items" setting is implemented.
- SMPP: the Recipient setting is implemented.
- PBXLeg: the MediaChannelOption function is implemented; it replaces a set of undocumented functions that are deprecated now.
- SIP: support for remote servers located behind far-end NAT has been added.
- SIP: URI "username-parameters" are supported now.
- CLI: the LISTMODULES, SETSTATELEMENT commands are implemented.
- CLI: the DATASET command is documented.
- CG/PL: the UPDATESTATISTICS procedure is deprecated, the STATISTICS function is implemented.
- CG/PL: the DATASETCREATE function is implemented.
- TLS: handshake records sent using several TLS low-level records are supported now.
- XIMSS: the "client-side" programming libraries for the XIMSS protocol are released.
- XIMSS: the datasetCreate, readConfigFile operations are implemented.
- XIMSS: the messageRedirect and messageForward command attributes are extended.
- XIMSS: the [rpop|rsip][List|Update] and balance operation documentation is updated.
- XIMSS: the messageRedirect, contactFind, submitMessage operations are extended.
- NETWORK: support for multi-homed NATed has been added to Session "Fixed IP Address" checks.
- Skins: sub-datasets (language.subname.data) files are supported now.
- PBXApp: the Referred-By field data is stored as a dictionary now.
- Billing: "charge when expired" functionality is implemented, the "total" result element is added.
- LIST: an option to preserve the original Message-IDs in the "feed" mode is implemented.
- SMTP: the 'client' value for the Advertise options now means "fixed clients" (i.e. without temporary clients)
- Bug Fix: CALENDAR: 5.4c1: if a Calendar contained a stand-alone item with an event exception, its Alarm could cause memory corruption
- Bug Fix: TLS: 5.4c1: TLS "resume sessions" connections to servers that did not support TLS resuming could result in crashes.
- Bug Fix: Calendar: 5.4c1: deleting all calendar items (ICS subscribe/import) could crash the server.
- Bug Fix: Calendar: 5.4c1: calendar items with attachments had an incorrect MIME structure.
- Bug Fix: SIP: 5.4c1: field parameters w/o value could make the entire SIP packet unreadable.
- Bug Fix: CLUSTER: 5.4c1: requests to NODEs on other cluster members could not be relayed.
- Bug Fix: HTTP: 5.4c1: when processing CGI requests, an incorrect Content-Length field was added to the response.
- Bug Fix: ACCOUNT: 5.4c1: OS-based passwords did not work.
- Bug Fix: MAILBOX: 5.4c1: MailDir-type mailbox reading could crash the server under heavy concurrent load.
- Bug Fix: CalDAV: 5.4c1: retrieved INBOX requests did not have the METHOD property.

**5.4c1** 17-Jan-2011

`Valid Core License Keys:` issued after 01-Jan-2010.

- External Mailbox storage functionality is implemented.

- SMPP: the Module is implemented (SMS send/receive).
- STUN: the Module is implemented.
- CardDAV: the Module is implemented.
- MAPI: the 64-bit version is included.
- Kernel: all string comparison operations support UTF-8 encoding.
- Kernel: string case-insensitive sortings place diacritics after their respective base letters.
- Foundation: object textual representations can contain comments.
- SMTP: the HELO Domain Prefix option is implemented.
- SMTP: the IPv6 DNS records are supported in reverse-checks now.
- QUEUE: the Enqueue Asynchronously supports more values now.
- Signal: Instant Message Logging is implemented.
- WebDAV: all WebDAV-based protocols (CalDAV, FileDAV) are reimplemented.
- WebDAV: the "principal" realm is implemented.
- WebDAV: the Access Control functionality (RFC3744) is implemented.
- GROUPWARE: recurrence exceptions are supported now.
- ACCOUNT: storage file locking is implemented.
- AIRSYNC: redesigned to support Seq-based Mailbox IDs.
- AIRSYNC: newer protocol versions are supported now (Android support).
- AIRSYNC: recurrence exceptions are supported now.
- WebUser: recurrence exceptions (display) are supported now.
- WebUser: the Contact Categories editor is implemented.
- CalDAV: recurrence exceptions are supported now.
- CalDAV: free-busy requests are supported now.
- CalDAV: subscriptions to calendar mailboxes in other Accounts are supported now.
- FileDAV: file locking is implemented, Windows and MacOS remote file systems are supported now.
- NETWORK: NAT Ping period settings for UDP and TCP clients are separated now.
- NETWORK: the "protocol error" settings for Temporarily Blocked Addresses is implemented.
- NETWORK: support for far-end "multihomed NATs" is implemented.
- NETWORK: the "Limit Connections per Address" Listener setting can be specified separately for client and non-client IP addresses.
- MAILBOX: the MailDir-type mailboxes now cache the last open message file handle.
- CG/PL: explicit variable declarations are implemented.
- CG/PL: C-style ( /* ... */ ) comments are supported now.
- CG/PL: the Calendar processing functions are implemented.
- CG/PL: the Cryptography functions are implemented.
- CG/PL: the Convert function is implemented.
- CG/PL: the ObjectClass function is implemented.
- XIMSS: the httpCall parameters are extended.
- XIMSS, WebUser: the size limit for the File Storage downloadable via the Session realm has been removed.
- XIMSS: the folderBrowse command accepts "uid" message sets now.
- XIMSS: the Data Export HTTP operations are implemented.
- HTTP: RFC2817 is implemented.
- TFTP: RFC2347, RFC2348, RFC2349 are implemented.
- Security: RFC5746 is implemented.
- Foundation: the AES cipher is implemented.
- TLS: RFC3268 (the RSA-AES-SHA cipher suites) is implemented.
- TLS: hello_request renegotiating requests are supported now.
- TLS: the size limit for handshake certificates has been removed.
- TLS: client-side session resuming is implemented.
- SIP: the deprecated "External Gateway" functionality has been removed.
- SIP: response delaying and client IP "banning" is implemented for locally-generated error codes.
- SIP: misformed packets are treated as "reported protocol errors" and can put the sender into the Temporarily Blocked Addresses list.
- Admin: when a Domain cannot be abandoned because some Accounts are in use, those Account names are

recorded in the Log.

- WebAdmin: the Account Setting pages are reorganized.
- QUEUE: the DSN (RFC3464) reports contain the Diagnostic-Code fields.
- ROUTER: the tel: URI routing is documented.
- CALLLEG: UPDATEs for pending outgoing INVITEs are supported now.

---

# Summary

**Foundation**

- now object textual representations may contain comments.
- the AES cipher is implemented.
- now timestamps in remote past (like year 1902) do not cause certificate parsing failures.
- a workaround for Solaris 11 realloc() bug is implemented.

**Kernel**

- all string comparison operations support UTF-8 encoding.
- string case-insensitive sortings place diacritics after their respective base letters.
- the --DelayOnStart startup option is implemented.
- the --ExcludeLocalIP startup option is implemented.
- 32-bit Unicode symbols are supported now.

**Pronto**

- Pronto Version 4 is included.
- The Pronto! HTML5 version is released.
- Pronto Mobile: Pronto! for Android is included.
- Pronto Mobile: Pronto! for iPhone has been released (via Apple Store).

**MAPI**

- the 64-bit version is included.
- the MAPI Connector version 1.54 is included.

**AIRSYNC**

- redesigned to support Seq-based Mailbox IDs.
- newer protocol versions are supported now (Android support).
- recurrence exceptions are supported now.
- when a PRIVATE item is created or modified, the Hidden flag is added to the item message.
- devices using the "validate" client-id (such as Android clients) are auto-provisioned now.
- the ItemOperation Fetch operation for retrieving a message MIME body is implemented.
- S/MIME compatibility with some Android clients is improved.
- some Unicode symbols are filtered out, as Apple devices crash when receiving those symbols.

**CalDAV**

- all WebDAV-based protocols (CalDAV, FileDAV) are reimplemented.
- the "principal" realm is implemented.
- the Access Control functionality (RFC3744) is implemented.
- Mozilla Lightning compatibility is improved.

**FileDAV**

- file locking is implemented, Windows and MacOS remote file systems are supported now.

**CalDAV**

- recurrence exceptions are supported now.
- free-busy requests are supported now.
- subscriptions to calendar mailboxes in other Accounts are supported now.

**CardDAV**

- CardDAV: the Module is implemented.

**DIRECTORY**

**ROUTER**

- the tel: URI routing is documented.

**Admin**

- when a Domain cannot be abandoned because some Accounts are in use, those Account names are recorded in the Log.

**LOG**

- the Source IP Address setting is implemented.

**NETWORK**

- NAT Ping period settings for UDP and TCP clients are separated now.
- the "protocol error" settings for Temporarily Blocked Addresses is implemented.
- support for far-end "multihomed NATs" is implemented.
- the "Limit Connections per Address" Listener setting can be specified separately for client and non-client IP addresses.
- support for multi-homed NATed has been added to Session "Fixed IP Address" checks.
- all operations that supported DNS A-type records now support the AAAA-type records, too.
- the DNR Source IP Address setting is implemented.
- the Relay for clients behind the same NAT option is implemented.
- the TCP Listener "connection aborted"-type errors are logged, but ignored now.

**CLUSTER**

- the relayed requests to Cluster-wide Local Units now search that Unit only.

**DOMAIN/ACCOUNT**

- storage file locking is implemented.
- the CHRONOS Tasks are implemented.

**MAILBOX**

- External Mailbox storage functionality is implemented.
- the MailDir-type mailboxes now cache the last open message file handle.

## QUEUE

- the Enqueue Asynchronously supports more values now.
- the DSN (RFC3464) reports contain the Diagnostic-Code fields.
- the pre-fetched subject size limit is increased to 2000 bytes.

## SIGNAL

- Access right checks for Event package subscriptions have been improved (see the Signal section of the manual)
- Offline Storage for Instant Messages is implemented.
- an Account Setting controlling Instant Messages delivery to SIP devices is implemented.
- an Account Preference controlling non-Roster senders of Instant Messages is implemented.
- Instant Message Logging is implemented.
- the "ActiveHold" workaround now works when a MOH channel is not created for that call leg.
- support for "collapsable proxies" is added to B2BUA calls made in the "connected" state.

## CALLLEG

- UPDATEs for pending outgoing INVITEs are supported now.

## PBXLEG

- the MediaChannelOption function is implemented; it replaces a set of undocumented functions that are deprecated now.
- the Referred-By field data is stored as a dictionary now.
- the Record function now records audio data from all media legs.
- the SendCallNotify function is implemented.

## MEDIAPROXY

- the UDP Port Restricted option now applies to non-NAT clients only.

## CALENDAR

- recurrence exceptions are supported now.
- if a VEVENT lacks the DTSTART property, it is added automatically.

## Security

- RFC5746 is implemented.

## HTTP

- RFC2817 is implemented.
- the RootDAV and UserDAV Log-Level settings are implemented.

## SIP

- the deprecated "External Gateway" functionality has been removed.
- response delaying and client IP "banning" is implemented for locally-generated error codes.
- misformed packets are treated as "reported protocol errors" and can put the sender into the Temporarily Blocked Addresses list.
- support for remote servers located behind far-end NAT has been added.
- URI "username-parameters" are supported now.
- URI parsing/composing is improved.
- the 503-type response is processed as a regular one if there is no more addresses to try; this is done to preserve

the response Reason: field.

## FTP

- the SITE NAMEFMT and SITE LISTFMT commands are implemented now (OS/400 client support).

## TFTP

- RFC2347, RFC2348, RFC2349 are implemented.

## SMTP

- the HELO Domain Prefix option is implemented.
- the IPv6 DNS records are supported in reverse-checks now.
- the 'client' value for the Advertise options now means "fixed clients" (i.e. without temporary clients)
- outgoing STARTTLS connections now use both the TLSv1 and SSLv3 protocols.

## MAILLIST

- an option to preserve the original Message-IDs in the "feed" mode is implemented.

## XMPP

- the XEP-114 extension is implemented (support for IM gateways).
- the XEP-0203 (Delayed Delivery) and XEP-0160 (Offline Messages) are implemented.
- the "Supplementary Discovery Items" setting is implemented.
- secure (SSL/TLS) server-server communication is implemented.

## SMPP

- the Module is implemented (SMS send/receive).

## WebAdmin

- the Account Setting pages are reorganized.

## WebUser

- the size limit for the File Storage downloadable via the Session realm has been removed.
- recurrence exceptions (display) are supported now.
- the Contact Categories editor is implemented.
- the mailbox menu elements now contain UTF-8 strings (instead of encoded UTF-7 strings).
- the Take Certificate operation does not store duplicates now.
- the S/MIME signing operation accepts contacts with multiple X.509 certificates.

## XIMSS

- the "client-side" programming libraries for the XIMSS protocol are released.
- the httpCall parameters are extended.
- the size limit for the File Storage downloadable via the Session realm has been removed.
- the folderBrowse command accepts "uid" message sets now.
- the Data Export HTTP operations are implemented.
- the datasetCreate, readConfigFile operations are implemented.
- the messageRedirect and messageForward command attributes are extended.
- the [rpop|rsip][List|Update] and balance operation documentation is updated.
- the messageRedirect, contactFind, submitMessage operations are extended.

- the mode attribute for the signalBind operation is supported now.
- the presenceSet operation now supports custom status messages.
- the Session Time Limit is applied to the TCP-based sessions, too.
- the setTimeout command is implemented, the readStatus command returns the session timeout information.
- the signalBind operation "deviceName" attribute has been renamed into "clientID" (the old attribuite is accepted, too).
- file uploading using the HTTP PUT method is implemented.
- the fileStore operation has been extended.

## CLI

- the LISTMODULES, SETSTATELEMENT,PDATESCHEDULEDTASK, GETIPSTATE commands are implemented.
- the DATASET command is documented.

## Skins

- sub-datasets (language.subname.data) files are supported now.

## WSSP

- a quoted string can be used as an expression.

## CG/PL

- explicit variable declarations are implemented.
- C-style ( /* ... */ ) comments are supported now.
- the Calendar processing functions are implemented.
- the Cryptography functions are implemented.
- the Convert function is implemented.
- the ObjectClass function is implemented.
- the UPDATESTATISTICS procedure is deprecated, the STATISTICS function is implemented.
- the DATASETCREATE function is implemented.
- the IsWhiteSpaces function is implemented.
- optional parameters for the built-in functions are implemented.

## TLS

- RFC3268 (the RSA-AES-SHA cipher suites) is implemented.
- hello_request renegotiating requests are supported now.
- the size limit for handshake certificates has been removed.
- client-side session resuming is implemented.
- handshake records sent using several TLS low-level records are supported now.

## STUN

- the Module is implemented.

## Billing

- "charge when expired" functionality is implemented, the "total" result element is added.

# CommuniGate Pro

# Version 5.3 Revision History

RSS

Cluster upgrade note: for rolling upgrades, your Cluster must be upgraded to the 5.2.17 or better version before upgrading to any 5.3 version.

**5.3.15** 24-Oct-2011
`Valid Core License Keys:` issued after 01-Dec-2008.

- SMTP: the host name length limit is increased to 256 bytes.
- Bug Fix: CLUSTER: 5.1: in some special cluster configurations proxying of SMTP connections might not work properly.
- Bug Fix: ACCOUNT: 5.3: retrieving Account "special ACLs" when the Account settings were being modified could crash the server.
- Bug Fix: WebUser: 5.0: the "safe" redirection to external links might not work in Internet Explorer 8.
- Bug Fix: AIRSYNC: 5.3c1: when a client started by adding items to a mailbox, those items could be sent back to it as "new".
- Bug Fix: CG/PL: 5.2: the DIRECTORYSEARCH function could cause server crashes.
- Bug Fix: CG/PL: 5.1: the OBJECTTOJSON function produced non-standard "\e" symbols.
- Bug Fix: CG/PL: 5.1: the OBJECTTOJSON could produce unencoded XML text.
- Bug Fix: PBXApp: 5.0: the REMOTEURI and LOCALURI functions did not escape URIs.
- Bug Fix: PBXApp: 5.2: the value for Privacy header was not delivered to PBX applications.

**5.3.14** 02-Jun-2011
`Valid Core License Keys:` issued after 01-Dec-2008.

- Pronto: the [Pronto version 3.14](#) is included.
- Kernel: the Russian TimeZones have been updated.
- Bug Fix: Kernel: converting to lowercase failed for the Greek leter "A".
- Bug Fix: XIMSS: 5.3.4: address-type message header fields could be encoded into the Japanese charsets incorrectly.
- Bug Fix: WebUser: 5.3: Simplex-based skins might fail to display the folder tree.
- Bug Fix: AirSync: 5.3c2: sending time zones without daylight saving times could crash some devices (Android clients).
- Bug Fix: DOMAINS: 5.0: if CGatePro->LDAP renaming table changed only the lower/uppercase of some setting name, that setting was not stored in the Central Directory.

**5.3.13** 04-Mar-2011
`Valid Core License Keys:` issued after 01-Dec-2008.

- Kernel: the Africa/Namibia time zone has been added.
- Pronto: the [Pronto version 3.13](#) is included.
- Bug Fix: SIP: 5.3.5: packets coming from "banned" addresses were not dropped.
- Bug Fix: RPOP: 5.3c2: many concurrent connection failures could cause memory corruption.
- Bug Fix: XIMSS: 5.3c1: the mailboxACLUpdate operation could not set an empty right set.

**5.3.12** 17-Jan-2011
`Valid Core License Keys:` issued after 01-Dec-2008.

- SIP: the "tel:" schema processing is improved.
- GROUPWARE: Contact sorting is using the YOMI name VCard field if it is present.
- Bug Fix: CLUSTER: 5.3.5: NAT Site IP List data could not be updated in a Cluster.
- Bug Fix: CALDAV: 5.3c1: "newer item already exists" situations resulted in the error code 500 (instead of 412).
- Bug Fix: CALLLEG: 5.2c1: re-INVITE requests did not contain Min-SE header field when session timer was activated.
- Bug Fix: CLI: 5.3c1: the FINDACCOUNTSESSION command could not find sessions that have set HTTP cookies.

**5.3.11** 03-Dec-2010
`Valid Core License Keys:` issued after 01-Dec-2008.

- Pronto: the [Pronto version 3.11](#) is included.
- Kernel: Australian time zone data has been updated to reflect new daylight saving time periods.
- Signal: the OPTIONS request to an Account w/o any SIP registration is auto-confirmed.
- MEDIA: now "music-on-hold" is not ignored in mixers used as media relays.
- Platform: AIX: the Server executable can use larger data segments now.
- Kernel: Email addresses starting with the dot symbol are accepted now.
- AIRSYNC: deleting from the Trash mailbox is always processed as "permanent".
- Big Fix: NamedTasks: 5.3c2: attempts to create a Named Task for an unknown Account could crash the server.
- Big Fix: CLUSTER: 5.3c3: reading mailbox message attributes could fail.
- Big Fix: CLUSTER: 5.3c2: RSIP registrations were directed to the registering cluster member, rather to the entire Cluster.
- Bug Fix: CHRONOS: 5.3c1: after an Account is renamed, its Chronos tasks were not rescheduled.
- Bug Fix: Billing: 5.3c1: the result of the "reservation release" operation did not contain the reservation name.
- Bug Fix: XIMSS: 5.3c2: the messageAttrData data messages were misformed.

- Bug Fix: XIMSS: 5.3.8: the dateFrom attribute in the findEvents response was incorrect for all-day recurrent events.
- Bug Fix: SIP: 5.3.3: media proxy created in mid-dialog could get an incorrect "direction" flag.
- Bug Fix: POP: 5.2c1: when a message is deleted when it is being retrieved, the POP connection was not closed.

**5.3.10** 18-Oct-2010
`Valid Core License Keys:` issued after 01-Dec-2008.

- Pronto: the [Pronto version 3.10](#) is included.
- MAPI: the MAPI version 1.53.10.1 is included.
- XIMSS: now multi-address E-mail fields are composed as single RFC Header fields.
- Bug Fix: HTTP: 5.3.3: the module could incorrectly request authentication when trying to access a personal file storage.
- Bug Fix: CALLLEG: 5.2c1: responses to SDP-less UPDATE requests could lack session-expiration fields.
- Bug Fix: XIMSS: 5.3.9: the findEvents operation could return incorrect data for All-day Events.

**5.3.9** 30-Aug-2010
`Valid Core License Keys:` issued after 01-Dec-2008.

- Pronto: the [Pronto version 3.9](#) is included.
- Pronto: the MS Windows and MacOS X installers for the "AIR" version are included.
- Platform: HPUX: the FIOSNBIO ioctlsocket call is used to set the non-blocking socket mode.
- Bug Fix: WebAdmin: 5.1c1: if an Account Import file line had a missing Alias or TelNum field, the server could crash.
- Bug Fix: WebUser: 4.2: if a new Contact Group member had a Real name containing the quote mark, the composed VCardGroup was unreadable.
- Bug Fix: MEDIA: 5.1: misformed SDP audio media data could crash the server.
- Bug Fix: SECURITY: 3.1: the {SSHA} encrypted passwords could support only 8-byte salts.
- Bug Fix: CG/PL: 5.3c2: the MAILBOXUIDBYORIGUID function did not work.

**5.3.8** 15-Jun-2010
`Valid Core License Keys:` issued after 01-Dec-2008.

- Pronto: the [Pronto version 3.8](#) is included.
- SIP: if a session is started with a non-UDP request, the Record Routes added contain the transport parameter.
- Bug Fix: SIP: 5.2c2: multi-part packet contents were re-composed incorrectly if some of the parts contained binary zeros.
- Bug Fix: DOMAIN: 5.3.1: unique Named Task names were created using uppercase symbols, causing problems for the Apple iChat client.
- Bug Fix: SIGNAL: 5.3.7: the <composing /> stanza could be added to some IM requests.
- Bug Fix: SIGNAL: 5.3c2: when re-sending a REGISTER/OPTIONS request with credentials a bogus timeout could be set.
- Bug Fix: SIGNAL: 5.3c1: the "gr=" URI parameter was not supported for ACK and BENOTIFY requests.
- Bug Fix: SIPS: 5.2.1: if an old-style CANCEL request arrived before its INVITE request, its transaction could get stuck in the queue.
- Bug Fix: LOG: 5.1c2: External Logger records with the Facility Code larger than 11 were composed incorrectly.
- Bug Fix: SESSIONS: 5.3.2: when a session timed out and it was being closed, a deadlock could occur.
- Bug Fix: CALLLEG: 5.3.2: the REGISTER operation (used in RSIP) could use random timeout values which could cause various deadlocks and loops.
- Bug Fix: AUTH: Kerberos authentication did not work for secondary Domains.

**5.3.7** 10-May-2010
`Valid Core License Keys:` issued after 01-Dec-2008.

- Pronto: the [Pronto version 3.7](#) is included.

- LOG: now the TLS i/o data is recorded only when the TLS Log Level is set to All Info.
- SIGNAL: the avatar hash data is added to the distributed presence signals. Users need to update their profiles/avatars (via XMPP clients, Pronto! or WebMail) to enable this functionality.
- XMPP: XEP-0055 (Jabber Search) is implemented.
- XMPP: support for XEP-0085 (Chat State Notifications) is implemented.
- AIRSYNC: some Unicode symbols are filtered out as a workaround for the Apple iPhone parser bug.
- STATISTICS: the Log header line listing all elements are now stored at the beginning of each new Statistics Log file.
- Bug Fix: ACCOUNT: 5.3.6: Rule modifications on the Account owner behalf could cause memory corruption.
- Bug Fix: ACCOUNT: 5.2c1: using special mailbox names like $SENT$ pointing to non-ASCII names could fail in the UTF8-based environments.
- Bug Fix: MAILBOX: 5.3c2: if sync data was corrupted, it was not removed from the Account Info file.
- Bug Fix: CG/PL: 5.2c2: the "+" operator did not work for datablock and string arguments.
- Bug Fix: DIRECTORY: 3.1: search filters were ignored in "base"-type search operations.
- Bug Fix: UDPPROXY: 5.3c2: when a NATed client switched its media port, the proxy did not accept the new data source.
- Bug Fix: HELPERS: 5.3c3: the [PRE|POST]UPDATE External Authenticator helper program was not called for the "set"-type Account Settings operations.
- Bug Fix: XMPP: 5.3c1: if the information was to be sent to remote servers using two <presence> stanzas, only one stanza was sent.

**5.3.6** 27-Apr-2010
`Valid Core License Keys:` issued after 01-Dec-2008.

- Pronto: the Pronto version 3.6 is included.
- DOMAIN: the Auto-Create Chatrooms Domain setting is implemented.
- CHATROOMS: additional XEP-0045 functions are implemented (configuration/destroy requests, info query, voice requests, etc.)
- XMPP: IQ-Based Avatar storage is implemented.
- XMPP: "forbidden" response processing changed to provide a workaround for an Apple iChat looping bug.
- CLI: the [GET|SET][CLUSTER]PROTECTION commands are renamed into [GET|SET][CLUSTER]NETWORK commands (the old commands continue to work).
- HTTPOutput: non-absolute URLs in 3xx responses are supported now.
- DOMAIN: when a forwarder is removed, it is removed from all Groups in its Domain.
- Bug Fix: CALLLEG: 5.3c1: UPDATE and INFO requests received within early dialogs were rejected.
- Bug Fix: DIRECTORY: 5.3.2: using LDAP Provisioning with a Cluster-wide Local Unit could cause connection loops.
- Bug Fix: DIRECTORY: 5.3.2: Cluster-wide Local Units worked incorrectly if mounted not at the tree root.
- Bug Fix: XMPP: 5.1.2: the in-session iq requests (roster updates) did not have the id attribute.
- Bug Fix: WebAdmin: 5.3c1: the "flow"-type settings could be modified incorrectly.
- Bug Fix: ACCOUNT: 5.3c1: the Failed Logins Limit settings worked incorrectly.
- Bug Fix: SIGNAL: 5.3c2: an incorrect Contact address in XMPP-type requests could crash the server.
- Bug Fix: SIP: 5.3.1: the "Signer" Record-Route was not include into registration Path, causing problems for Microsoft RTC clients.
- Bug Fix: WebUser: 5.2c1: the Use Letter Charset preference did not work.
- Bug Fix: CLUSTER: 5.3c1: the Suspend/Resume Domain operations could fail in a Cluster.

**5.3.5** 30-Mar-2010
`Valid Core License Keys:` issued after 01-Dec-2008.

- Pronto: the Pronto version 3.5 is included.
- NETWORK: the NAT Site IP List is implemented.
- ACCOUNT: when the Account Archive Messages After / Delete Messages After settings are changed, the Account Chronos wakeup times are re-calculated.

- CALENDAR: Calendar ALARMs via E-mail are implemented
- CALENDAR: the Server-side Processing Account Preferences are documented.
- XMPP: XEP-0008, XEP-0012, XEP-0202 are implemented.
- HTTP: the MS Autodiscover protocol is implemented.
- SIP: the "privacy" extension is supported now.
- STATISTICS: the imapMAPIActive element is re-implemented.
- SMTP: the "disk is full" OS error is processed as a temporary problem.
- Bug Fix: CALENDAR: 5.3c3: when opening a remote (HTTP-retrieved) Calendar, items without the PUBLISH method specified were ignored.
- Bug Fix: Kernel: 5.2.4: when an external certificate was verified, the stock certificates were used incorrectly.
- Bug Fix: WSSP: 5.3.1: CALENDARTIMEMENU could display the 12:00AM item as "past".
- Bug Fix: SMTP: 5.3c1: the composed Received: header field contained "short" authenticated Account names (i.e. the Main Domain name was not included).
- Bug Fix: WebAdmin: 5.3c2: authentication using a simple username did not work if the IP address was assigned to a secondary Domain.
- Bug Fix: WebAdmin: 5.3c1: the AirSyncAllowed data in Import files was processed incorrectly.
- Bug Fix: XIMSS: 5.3c1: Auto-login did not work in Clusters.
- Bug Fix: WebUser: 4.3.1: the Event composer page incorrectly displayed "Availability" Free/Busy data when the daylight saving status of the displayed and the current time were different.
- Bug Fix: WebUser: 5.2.10: the HiddenSkins setting could not be used to hide the "unnamed" (Basic) Skin.

### 5.3.4 09-Mar-2010
`Valid Core License Keys:` issued after 01-Dec-2008.

- Pronto: the Pronto version 3.4 is included.
- Platform: the Windows/AMD64 version is released.
- WebUser: plain text space formatting is improved ("ASCII-tables" are supported now).
- WebUser: now the Send Requests option is disabled by default if an Event is composed not in the Default Calendar mailbox.
- CLI: the SETACCOUNTPASSWORD command supports the optional CHECK keyword now.
- WebAdmin: the Temporarily Blocked Addresses -> Block After setting is documented.
- AIRSYNC: the MIMETruncation parameter is supported now.
- Admin: now the External AUTH helper application receives the name of the Account performing the provisioning operations.
- Admin: the HiddenProntoModules Account setting is implemented.
- RSIP: the optional Target field is supported now.
- Bug Fix: CLI: 5.2c1: incorrect parameters for the GETACCOUNTSTAT and RESETACCOUNTSTAT commands could crash the server.
- Bug Fix: ACCOUNT: 5.3.3: if an event segment did not have expiration time set, the segment expired immediately (Message Waiting Indicator did not work).
- Bug Fix: Foundation: 5.3c1: "bare" XML in inter-cluster communication could contain unencoded EOL symbols, causing errors and protocol de-synchronization.
- Bug Fix: NamedTasks: 5.3c3: task names were processed as case-sensitive, causing problems for come applications (such as IM chatrooms).
- Bug Fix: Admin: 5.3c1: the new Access Rights caching mechanism caused "delayed updates" on the WebAdmin "Access Rights" page.
- Bug Fix: SIP: 5.3.3: processing REGISTER requests without the Expires field could cause server crashes under heavy load.
- Bug Fix: DOMAIN: 5.3c2: Named Task search was case-sensitive, causing Multi-user Chat problems.
- Bug Fix: GROUP: 4.0: the Expand Group option could leave some duplicated addresses in the final recipient list.
- Bug Fix: MAILBOX: 5.2c1: hard-links were never used for messages in MailDir mailboxes.

### 5.3.3 25-Feb-2010
`Valid Core License Keys:` issued after 01-Dec-2008.

- WSSP: the JSON text element is implemented.
- Pronto: the [Pronto version 3.3](#) is included.
- MAPI: the MAPI version 1.53.3.2/1.52.53.1 is included.
- SPELLCHECK: UTF-8 "symbol" offsets are supported.
- ACCOUNT: when a File Storage folder is created, it gets a copy of the outer folder ACL.
- WEBUSER: FireFox "x-dns-prefetch-control" meta-tag is added to disable the spammers DNS-based "harvesting" techniques.
- MEDIA: DTMF processing now supports more devices incorrectly using 'new' and 'end' RFC2833 flags.
- MEDIA: NAT stream processing improved to support mixer peers behind the same NAT/Firewall.
- SIP: when an SDP element is received inside a multipart body, it is relayed together with other body parts (SIP-T support).
- HTTP: the Range request field is supported now.
- ACCOUNT: call dialogs initiated before a server restart or before a full cluster restart are auto-removed.
- CALENDAR: the "import" operation now ignores 'recurrence exception' items.
- Bug Fix: WebAdmin: 5.3.2: the Settings Signal page Update operation worked, but displayed the Incorrect Parameter Value error.
- Bug Fix: CLUSTER: 5.2c1: XML SDP composing procedure did not use the "altAddress" attribute, causing media transfer problems for XIMSS clients.
- Bug Fix: CLUSTER: 5.3c2: Calendar alarm time was updated incorrectly in virtual (inter-cluster) Mailboxes.
- Bug Fix: CLUSTER: 5.3c2: inter-cluster file operations could incorrectly pass the authentication data.
- Bug Fix: PBXApp: 5.3c2: updating SDP in the "bridged" mode did not store the new SDP, resulting in incorrect "SDP has not been changed" situations.
- Bug Fix: CG/PL: 5.2c1: the SetCaseSensitive procedure worked incorrectly.
- Bug Fix: ADMIN: 5.3c1: the "Insert All" operation did not add Named Task records to the Central Directory.
- Bug Fix: ADMIN: 5.3c1: when an Account was deleted, its Named Tasks were not removed.
- Bug Fix: AIRSYNC: 5.2: when the Sent Mailbox preference was set to an empty string, no message could be submitted.
- Bug Fix: DIRECTORY: 3.0: Local Unit modification records longer than 32K could not be read.
- Bug Fix: DIRECTORY: 5.3c2: the Local Unit Schema missed some attributes.
- Bug Fix: XMPP: 5.2.5: when a message request contains two "body" elements, none could reach the destination.
- Bug Fix: XIMSS: 5.2c1: the directory name in file operation responses could be sent without XML string escape symbols.
- Bug Fix: SIP: 5.3c1: in-dialog requests with "gr" (gruu) URI parameter were rejected.
- Bug Fix: STATISTICS: 5.3c3: "failedAUTHs" and "successfulAUTHs" element values were calculated incorrectly.

**5.3.2** 26-Jan-2010
`Valid Core License Keys`: issued after 01-Dec-2008.

- Pronto: the [Pronto version 3.2](#) is included.
- DIRECTORY: Cluster-wide Units are implemented.
- AIRSYNC: UTF8 data-cleaning is implemented for Calendar, Tasks, and Contacts data (malformed UTF8 data causes Apple iPhone crashes).
- HTTP: "chunked" requests are supported now.
- CLI: the Directory Administration commands are implemented.
- CLUSTER: now NOTIFY Signal requests are clustered: they are sent via the Cluster Members with the Signal service enabled.
- XIMSS: the datasetListSubset operation is implemented.
- CG/PL: the DATASETLISTSUBSETS function is implemented.
- ACCOUNT: the Calls and NOTIFY flow control settings are moved to Account settings.
- SIP: symbolic (FQDN) "sentby" Via fields are supported now.
- Bug Fix: CALLLEG: 5.1.2: the Replaces "method" parameter was checked incorrectly.
- Bug Fix: CalDAV: 5.3.1: Dropbox error logging could crash the server.

Bug Fix: CalDAV: 5.3c2: SMTP ATRN polling could crash the server.
- Bug Fix: QUEUE: 5.3c3: messages sent to the "null" address were rejected with the "address is blacklisted" error code.
- Bug Fix: SIGNAL: 5.3.1: REGISTER requests with expires=0 Contact parameter were rejected.
- Bug Fix: AIRSYNC: 5.3c3: logging of certain Sync data modifications could crash the server.

**5.3.1** 12-Jan-2010
`Valid Core License Keys:` issued after 01-Dec-2008.

- Pronto: the [Pronto version 3.1](#) is included.
- Kernel: if a MIME part does not have Content-Type specified, the Content-Type is set using the MIME part file name suffix (if it exists).
- HTTP: access to private directories requires authentication only if non-authenticated (guest) access was denied.
- WebUser: the Chatroom Log viewer is implemented.
- WebUser: the Contact Editor now supports the Yomi Company Name and the Yomi Name fields.
- CalDAV: the "DropBox" functionality is supported now.
- CalDAV: the VAVAILABILITY requests are supported now.
- Chatrooms: semi-anonymous rooms implemented, iq relaying is implemented.
- MAILBOX: the Mailbox seqNum is assigned automatically, if absent, whenever the Mailbox meta-data is read.
- MEDIAPROXY: now separate counters are used to generate SDES PING packets for both media leg.
- Bug Fix: SIGNAL: 4.3c1: minimum and maximum REGISTER period limits were not applied to the "expires" Contact parameters.
- Bug Fix: CHRONOS: 5.3c2: the services was starting too late, and RPOP records converted from the old RPOP Domain settings were not scheduled.
- Bug Fix: MAILBOX: 5.3.0: large messages (> 3MB) could not be copied.
- Bug Fix: MAILBOX: 5.3c1: mailbox Seq info-element was checked incorrectly, resulting in repeated "seq restoring" attempts for some mailboxes.
- Bug Fix: RPOP: 5.3.0: failed connections caused extremely long delays (till the year 2050).
- Bug Fix: CalDAV: 5.3c4: the Apple iCAL client broke when listing calendars, if one of them did not have the 'color' attribute set.
- Bug Fix: CALENDAR: 5.3c1: incorrect "next recurrence" calculations could cause notification loops.
- Bug Fix: WebUser: 5.3c1: Contact Groups did not work in Address Books.
- Bug Fix: CG/PL: 5.1c1: the DateByYearDay function returned a result for a previous year.
- Bug Fix: CG/PL: 5.3c1: the MailboxOrigUIDByUID function returned an incorrect result.

**5.3.0** 28-Dec-2009
`Valid Core License Keys:` issued after 01-Dec-2008.

- Pronto: the [Pronto version 3.0](#) is included.
- MEDIA: the upsampling and downsampling algorithms are improved.
- AIRSYNC: additional provisioning attributes are supported now.
- CALDAV: the getcontenttype property is supported now (Lightning compatibility).
- HTTP: RFC4709 is implemented.
- XIMSS: the datasetRemove operation can process foreign datasets now.
- WSSP: the JAVASCRIPT: text element now escapes the < symbols (a workaround for browser bugs).
- MAILBOX: message attributes are copied with the message.
- CLUSTER: improved processing of the "Auto" value for the SIP Farm, Queue Submit, HTTP Client, POP Client settings.
- CLUSTER: message attributes are supported in virtual mailboxes now.
- CLUSTER: SIGNAL processing can be partially delegated to certain cluster members (frontends) now.
- GROUPWARE: the X-FirstName and X-LastName fields are added to VCard message items.
- PBXApp: the DisconnectEx procedure is implemented.
- CG/PL: the MAILBOXCOPY function parameters are extended.
- XIMSS: the messageAdded message can be sent in response to messageAppend and messageCopy requests.

- MAILLIST: symbol combinations (macro) can specify fixed-length numbers now.
- Bug Fix: Kernel: 5.3c1: some operations incorrectly checked Domain-level Access Rights.
- Bug Fix: SIP: 5.3c3: the Replaces header was processed incorrectly.
- Bug Fix: Calendar: 5.3c1: some monthly and yearly recurrence calculations were performed incorrectly.
- Bug Fix: DOMAIN: 5.3c1: INBOX-only accounts did not work.
- Bug Fix: AIRSYNC: 5.3c3: directory search did not compose the 'mail' attribute if it was absent.

---

**5.3c4** 17-Dec-2009
`Valid Core License Keys:` issued after 01-Dec-2008.

- Foundation: UTF8 case-converting utilities process more Roman, Greek, and Cyrillic symbols now.
- Foundation: IPv6 support for Windows is implemented (Windows Vista or newer is required).
- Kernel: Punycode (RFC3492) is implemented.
- ACCOUNT: Kerberos and TLS Certificate authentication methods now process the supplied usernames with the Router (eased security limitation).
- Email: support for IDN (international domains) is added to most E-mail modules and components.
- DOMAIN: international (non-ASCII) Domain Alias names can be used now.
- RPOP: the PROP component redesigned using the Chronos component. The RPOP record settings are moved from Domain to Account level.
- RPOP: the Mailbox setting is implemented to support direct delivery of retrieved messages.
- CLUSTER: the RPOP Client setting is implemented to allow the RPOP sessions to take place on selected servers only (usually - on frontends).
- CLI: the [GET|SET]ACCOUNTRPOP commands are removed. The [GET|SET]ACCOUNTRPOPS commands are implemented.
- CG/PL: now the comparison operations can be applied to strings and datablocks.
- HTTPOutput: the 1xx server responses are supported now.
- MEDIA: variable-frequency wave buffers are implemented (can be used with wideband codecs)
- LIST: addresses copied with the Copy as Cc option are enclosed into the angle brackets now.
- TFTP: the file write operations are supported now.
- LDAP: the Router Subtree feature is implemented.
- AIRSYNC: the Directory search operations use the user domain DN as the search base.
- Bug Fix: CLUSTER: 5.3c3: failover process could detect false controller conflicts.
- Bug Fix: FILES: 5.3c1: file storage size limitations were checked incorrectly.
- Bug Fix: ACCOUNT: 5.3c1: INBOX-only accounts could not be created.
- Bug Fix: AIRSYNC: 5.3c1: in some cases device re-synchronization did not send all mailbox items to the device.
- Bug Fix: RULES: 5.3c3: sending Instant Messages from Server/Cluster-wide Rules could crash the server.
- Bug Fix: DOMAIN: 5.3c1: the Domain-level Enabled Services and Login Method settings did not work.

**5.3c3** 24-Nov-2009
`Valid Core License Keys:` issued after 01-Nov-2008.

- Pronto: the [Pronto version 3.0c3](#) is included.
- Kernel: RFC3676 is implemented.
- RSIP: the Remote SIP component is implemented (see the Signal PSTN section of the details). The SIP Gateways manager is deprecated and it is prepared for removal.
- WebUser: the RSIP Signal settings page is implemented.
- CLI: the [GET|SET]ACCOUNTRSIPS commands are implemented.
- XIMSS: the rsipList operation is implemented.
- HTTPOutput: POST requests now support the x-www-form-urlencoded format.
- HTTPOutput: the HTTP Client Manager settings have been added to the Others page of the WebAdmin Interface.
- HTTPOutput: the dictionary-type "simple values" for POST fields are supported now.

- SIP: the Record-Route comparison routine is improved.
- DOMAIN: new Domain and Account statistics elements have been added.
- WSSP: the INCLUDE elements now accepts an expression as a filename.
- SIGNAL: requests directed (routed) to the NULL-address are completed immediately, returning the "OK" code.
- CLUSTER: the Controller now always reads the Heartbeat file and quits if some other server rewrites it.
- XIMSS: the httpCall "field" elements can contain non-string data now.
- WebAdmin: additional Local Account Manager settings are implemented.
- Bug Fix: MAILLIST: 5.3c1: HTTP access authorization did not work.
- Bug Fix: MAILLIST: 5.3c1: WebUser "Approve" operation did not work from the mailbox page.
- Bug Fix: WebADMIN: 5.3c1: Mailbox names could not be modified on the All-Server/All-Cluster Account Preferences pages.
- Bug Fix: AIRSYNC: 5.3c1: some devices could not sync.

**5.3c2** 09-Nov-2009
`Valid Core License Keys:` issued after 01-Nov-2008.

- CALENDAR: the "publish" operations preserve the original ("permanent") UID.
- Security: more built-in "trusted" certificates are added.
- XIMSS: the messageSubmit operation now copies the "Bcc" field data into the saved message.
- STATISTICS: larger threshold values can be specified now.
- XMPP: output channel statistic elements have been implemented.
- Kernel: when storing multi-line "object" (dictionary, array) data, data elements can be split to avoid creating very long text lines.
- XIMSS: when reading messages, the parts with "attachment" disposition are not included into message XML.
- XIMSS: */xml and */*+xml message parts are supported now.
- SIP: an "edge proxy" now supports calls from a far-end NAT'ed client calling a near-end NAT'ed client via the same server.
- WebUser: the Attach Profile vCard message composer option is implemented.
- DOMAIN: when creating new objects, reserved names (such is NULL, ERROR, ALL, etc.) are rejected now.
- ACCOUNT: the local delivery/sending Flow Control setting have been moved to Account settings.
- ACCOUNT: the Failed Login Limit setting have been moved to Account settings.
- XMPP: the XMPPS and XMPPI System Log tags are swapped (XMPPS is used for client sessions now).
- WebAdmin: the "Pronto" section has been added to the Account Preference page.
- Bug Fix: CALENDAR: 5.3c1: alert times could be calculated incorrectly, resulting in server loops.
- Bug Fix: XIMSS: 5.3c1: copying file attributes could crash the server.
- Bug Fix: CLUSTER: 5.3c1: starting a NamedTask in a shared Domain could cause inter-server communication deadlocks.

**5.3c1** 1-Nov-2009
`Valid Core License Keys:` issued after 01-Nov-2008.

- Chronos "daemon processing" is implemented.
- Pronto: the [Pronto version 3.0c1](#) is included.
- MAILBOX: "permanent UIDs" are implemented.
- MAILBOX: Message Attributes are implemented.
- ACCOUNT: the Archive Messages after and Delete Messages after settings are implemented.
- ACCOUNT: File Storage file attributes are implemented.
- ACCOUNT: when calculating number and total size of storage files, the ".meta" files are ignored now.
- ACCOUNT: File Storage ACLs are implemented.
- ACCOUNT: File Storage Subscription is implemented.
- ACCOUNT: the File Storage "pubcal" virtual directory (for calendar publishing) is implemented.
- ACCOUNT: Call Logs are now stored in GMT terms.
- ACCOUNT: the null@null identifier is implemented to grant access to unauthenticated users (guests).
- ACCOUNT: the Log Login/Logout setting is implemented.

- ACCOUNT: the Login log records are not created if the login operation is rejected because of a disabled service.
- ACCOUNT: the legacy "Netscape roaming" storage is deprecated. The /settings/ requests are now directed to the private/roaming/ File Storage directory.
- ACCOUNT: now the External Authenticator is consulted when an unknown object is met in non-routing operations, too.
- DOMAIN: Named Tasks are implemented.
- DOMAIN: the Fast Storage Type setting is implemented.
- AIRSYNC: synchronization logic has been changed to use the new "permanent UIDs" Mailbox feature.
- AIRSYNC: if an updated vCard object is equal to the existing one, the existing one is kept.
- XMPP: the module is re-implemented.
- XMPP: multiparty chat (XEP-0045) is implemented. See the Named Tasks section.
- XMPP: vcard-temp (XEP-0054) is implemented.
- XMPP: XMPP Ping (XEP-0199) is implemented.
- HTTP: WebDAV access to Account File Storage is implemented.
- TLS: outgoing SSL/TLS connections now use the Main Domain SSL/TLS Certificate if the remote server requests a certificate.
- TLS: client certificate verification is now supported for the old SSL (TLS v0) protocol, too.
- TLS: the Accept SSLv2 connections and Process Target Domain extensions TLS options are implemented.
- IMAP: the APPEND extensions are documented.
- PBX: the Click-to-Call functionality re-implemented using a CG/PL "clicktocall.sppr" task, and it became available in a Cluster environment.
- SIP: the GRUU (RFC draft) functionality is implemented.
- Foundation: XML objects are re-implemented to improve performance and namespace handling.
- DNR: the Balance Server Load and Use Supplementary Responses options are implemented.
- SMTP: the RBL "TXT" records are supported now.
- QUEUE: the Log Message Delivery setting is documented.
- QUEUE: the X-Auto-Response-Suppress header field does not mark a message as auto-generated.
- STATISTICS: PWD, AIRSYNC statistics elements are implemented.
- STATISTICS: The Statistics Log (storing all Statistic Elements data periodically) is implemented.
- SIGNAL: now type-02 CDR records are not generated for "retriable" errors.
- LOGS: the Supplementary Logs are documented.
- LDAP: the "extended match" operation is supported.
- Intercept: the "Deleted Mail" option implemented, the report message format changed.
- Skins: the "Classic" skin is retired.
- WebUser: the Kazakh language has been added.
- WebUser: the ';' symbol is accepted as a To/Cc/Bcc address list separator.
- WebUser: the Billing and Mobile Manager pages are implemented.
- WebUser: the Personal Info (vCard Profile) editor page is implemented.
- Admin: if an administration operation is performed by an administrator of a secondary domain, the full account name is recorded.
- HTTPOutput: the HTTP Client interface is documented, CG/PL and XIMSS documentation for the HTTP Call operations is updated.
- HTTPOutput: the supplFields request element is implemented.
- HTTPOutput: the connection time out is set now (to the same value as the transaction timeout).
- MEDIA: the Inband DTMF Signal/Noise Ratio setting is implemented.
- CALLLEG: INVITE-replaces for a REFERed INVITE are supported now (blind-transferred called being picked up).
- SDP: the XML presentation includes arbitrary attributes now.
- CG/PL: the "for-loop" operator is implemented.
- CG/PL: qualified external code names (moduleName::codeName) are implemented.
- CG/PL: the builtin FUNCCALL function and the PROCCALL procedure are implemented.
- CG/PL: the UPDATESTATISTICS procedure is implemented.
- CG/PL: the RADIUSCall function parameters are extended.
- CG/PL: the MAILBOXCOPYBYUID and MAILBOXSETFLAGSBYUID function renamed to

COPYMAILBOX and MAILBOXSETFLAGS (old names are preserved), function parameters are extended.

- CG/PL: the [READ|WRITE|APPEND|RENAME|DELETE|LIST]SITE[FILE|DIRECTORY] functions are renamed into [READ|WRITE|APPEND|RENAME|DELETE|LIST]STORAGE[FILE|DIRECTORY].
- CG/PL: the [READ|WRITE]STORAGEFILEATTR functions are implemented.
- CG/PL: the [GET|UPDATE]MESSAGEATTR functions are implemented.
- CG/PL: the LISTSITEFILES function result is changed (the elements are always dictionaries, the MetaModified timestamps are added).
- CG/PL: the SENDINSTANTMESSAGE function is extended, the SENDXMPPIQ function is implemented.
- CG/PL: the READENVIRFILE function is documented.
- CG/PL: the MAILBOX[ORIGUIDBYUID|UIDBYORIGUID] functions are implemented.
- PBXApp: the SETLOCALCONTACTPARAMETER procedure is implemented.
- PBXApp: the Alert-Info SETCALLPARAMETERS dictionary element is supported now.
- PBXApp: the SETINTERRUPTONDTMF function now controls DTMF interrupts for both audio playing and audio recording operations.
- PBXApp: Instant Message and XMPP IQ receiving, the ISINSTANTMESSAGEEVENT and ISXMPPIQEVENT functions are implemented.
- XIMSS: the retrieveURL operation is renamed into httpCall (the old name is still available).
- XIMSS: the reqSeq parameter is documented.
- XIMSS: the listUploaded operation is implemented.
- XIMSS: the newUploaded data message is implemented.
- XIMSS: the makeCall function now completes immediately, the Click-to-Call status reports come as asynchronous data messages now.
- XIMSS: the fileAttr[Read|Write], fileSub[List|Update] operations are implemented, the timeAttrModified attribute is added to the fileInfo data messages.
- XIMSS: the file[Read|Write] type attribute with "vcard" value is supported now.
- XIMSS: the iqSend/iqRead operations are implemented.
- XIMSS: the sendIM/readIM attributes are extended.
- XIMSS: the messageAttr[Read|Write] operations are implemented
- CLI: the GETSNMPELEMENT operation renamed into GETSTATELEMENT, its result format is changed to Number.
- CLI: the *WEBFILE* operations are renamed into *STORAGFILE* operations.
- CLI: the [READ|UPDATE]STORAGEFILEATTR operations are implemented.
- CLI: the GETNEXTSTATNAME, [GET|SET]MEDIASERVERSETTINGS commands are implemented.
- CLI: the [GET|SET]ACCOUNTSUBSCRIPTION commands are renamed into [GET|SET]MAILBOXSUBSCRIPTION.
- CLI: the [GET|SET]FILESUBSCRIPTION commands are implemented.
- CLI: the *NAMEDTASK commands are implemented.

# Summary

**Foundation**

- XML objects are re-implemented to improve performance and namespace handling.
- UTF8 case-converting utilities process more Roman, Greek, and Cyrillic symbols now.

**Kernel**

- Chronos "daemon processing" is implemented.
- RFC3676 is implemented.
- Punycode (RFC3492) is implemented.
- when storing multi-line "object" (dictionary, array) data, data elements can be split to avoid creating very long text lines.

if a MIME part does not have Content-Type specified, the Content-Type is set using the MIME part file name suffix (if it exists).
- Australian time zone data has been updated to reflect new daylight saving time periods.
- Email addresses starting with the dot symbol are accepted now.

## Platform

- the Windows/AMD64 version is released.
- Windows: IPv6 support is implemented (Windows Vista or newer is required).
- AIX: the Server executable can use larger data segments now.
- HPUX: the FIOSNBIO ioctlsocket call is used to set the non-blocking socket mode.

## Pronto

- the Pronto version 3.11 is included.
- the MS Windows and MacOS X installers for the "AIR" version are included.

## MAPI

- the MAPI version 1.53.10 is included.

## AIRSYNC

- synchronization logic has been changed to use the new "permanent UIDs" Mailbox feature.
- if an updated vCard object is equal to the existing one, the existing one is kept.
- the Directory search operations use the user domain DN as the search base.
- additional provisioning attributes are supported now.
- UTF8 data-cleaning is implemented for Calendar, Tasks, and Contacts data (malformed UTF8 data causes Apple iPhone crashes).
- the MIMETruncation parameter is supported now.
- some Unicode symbols are filtered out as a workaround for the Apple iPhone parser bug.

## CalDAV

- the getcontenttype property is supported now (Lightning compatibility).
- the "DropBox" functionality is supported now.
- the "VAVAILABILITY" requests are supported now.

## DIRECTORY

- Cluster-wide Units are implemented.

## Admin

- the Calls and NOTIFY flow control settings are moved to Account settings.
- the "Deleted Mail" Intercept option implemented, the Intercept report message format changed.
- if an administration operation is performed by an administrator of a secondary domain, the full account name is recorded.

## LOG

- the Supplementary Logs are documented.
- now the TLS i/o data is recorded only when the TLS Log Level is set to All Info.

## NETWORK

- the NAT Site IP List is implemented.
- the DNR Balance Server Load and Use Supplementary Responses options are implemented.

## CLUSTER

- the RPOP Client setting is implemented to allow the RPOP sessions to take place on selected servers only (usually - on frontends).
- improved processing of the "Auto" value for the SIP Farm, Queue Submit, HTTP Client, POP Client settings.
- the Controller now always reads the Heartbeat file and quits if some other server rewrites it.
- SIGNAL processing can be delegated to certain cluster members (frontends) now.

## DOMAIN/ACCOUNT

- the Archive Messages after and Delete Messages after settings are implemented.
- File Storage file attributes are implemented.
- when calculating number and total size of storage files, the ".meta" files are ignored now.
- File Storage ACLs are implemented.
- File Storage Subscription is implemented.
- Named Tasks are implemented.
- the File Storage "pubcal" virtual directory is implemented.
- international (non-ASCII) Domain Alias names can be used now.
- the null@null identifier is implemented to grant access to unauthenticated users (guests).
- the Log Login/Logout setting is implemented.
- the Login log records are not created if the login operation is rejected because of a disabled service.
- the legacy "Netscape roaming" storage is deprecated. The /settings/ requests are now directed to the private/roaming/ File Storage directory.
- now the External Authenticator is consulted when an unknown object is met in non-routing operations, too.
- the Fast Storage Type Domain setting is implemented.
- the local delivery/sending Flow Control setting have been moved to Account settings.
- the Failed Login Limit setting have been moved to Account settings.
- Kerberos and TLS Certificate authentication methods now process the supplied usernames with the Router (eased security limitation).
- Call Logs are now stored in GMT terms.
- when creating new objects, reserved names (such is NULL, ERROR, ALL, etc.) are rejected now.
- new Domain and Account statistics elements have been added.
- call dialogs initiated before a server restart or before a full cluster restart are auto-removed.
- now the External AUTH helper application receives the name of the Account performing the provisioning operations.
- the HiddenProntoModules Account setting is implemented.
- the Auto-Create Chatrooms Domain setting is implemented.

## MAILBOX

- "permanent UIDs" are implemented.
- Message Attributes are implemented.
- the Mailbox seqNum is assigned automatically, if absent, whenever the Mailbox meta-data is read.

## QUEUE

- support for IDN (international domains) is added to most E-mail modules and components.
- the Log Message Delivery setting is documented.
- the X-Auto-Response-Suppress header field does not mark a message as auto-generated.

## PBX

- the Click-to-Call functionality re-implemented using a CG/PL "clicktocall.sppr" task, and it became available in a Cluster environment.

## SIGNAL

- now type-02 CDR records are not generated for "retriable" errors.
- requests directed (routed) to the NULL-address are completed immediately, returning the "OK" code.
- the avatar hash data is added to the distributed presence signals. Users need to update their profiles/avatars (via XMPP clients, Pronto! or WebMail) to enable this functionality.
- the OPTIONS request to an Account w/o any SIP registration is auto-confirmed.

## CALLLEG

- INVITE-replaces for a REFERed INVITE are supported now (blind-transferred called being picked up).

## MediaServer

- the Inband DTMF Signal/Noise Ratio setting is implemented.
- variable-frequency wave buffers are implemented (can be used with wideband codecs)
- the upsampling and downsampling algorithms are improved.
- the SDP XML presentation includes arbitrary attributes now.
- DTMF processing now supports more devices incorrectly using 'new' and 'end' RFC2833 flags.
- NAT stream processing improved to support mixer peers behind the same NAT/Firewall.
- now "music-on-hold" is not ignored in mixers used as media relays.

## MediaProxy

- separate counters are used to generate SDES PING packets for both media leg.

## CALENDAR

- the "publish" operations preserve the original ("permanent") UID.
- the "import" operation now ignores 'recurrence exception' items.

## CONTACTS

- the X-FirstName and X-LastName fields are added to VCard message items.

## Security

- more built-in "trusted" certificates are added.

## TLS

- outgoing SSL/TLS connections now use the Main Domain SSL/TLS Certificate if the remote server requests a certificate.
- client certificate verification is now supported for the old SSL (TLS v0) protocol, too.
- the Accept SSLv2 connections and Process Target Domain extensions TLS options are implemented.

## HTTP

- WebDAV access to Account File Storage is implemented.
- RFC4709 is implemented.
- access to private directories requires authentication only if non-authenticated (guest) access was denied.
- "chunked" requests are supported now.
- the Range request field is supported now.

## HTTPOutput

- the HTTP Client interface is documented, CG/PL and XIMSS documentation for the HTTP Call operations is updated.
- the supplFields request element is implemented.
- the connection time out is set now (to the same value as the transaction timeout).
- POST requests now support the x-www-form-urlencoded format.
- the HTTP Client Manager settings have been added to the Others page of the WebAdmin Interface.
- the dictionary-type "simple values" for POST fields are supported now.
- the 1xx server responses are supported now.

## SIP

- the SIP Gateways manager is deprecated and it is prepared for removal.
- the GRUU (RFC draft) functionality is implemented.
- the Record-Route comparison routine is improved.
- an "edge proxy" now supports calls from a far-end NAT'ed client calling a near-end NAT'ed client via the same server.
- when an SDP element is received inside a multipart body, it is relayed together with other body parts (SIP-T support).
- if a session is started with a non-UDP request, the Record Routes added contain the transport parameter.

## RSIP

- the Remote SIP component is implemented (see the Signal PSTN section of the details).

## IMAP

- the APPEND extensions are documented.

## TFTP

- the file write operations are supported now.

## SMTP

- the RBL "TXT" records are supported now.

## RPOP

- the PROP component redesigned using the Chronos component. The RPOP record settings are moved from Domain to Account level.
- the Mailbox setting is implemented to support direct delivery of retrieved messages.

## MAILLIST

- symbol combinations (macro) can specify fixed-length numbers now.
- addresses copied with the Copy as Cc option are enclosed into the angle brackets now.

## STATISTICS

- PWD, AIRSYNC statistics elements are implemented.
- The Statistics Log (storing all Statistic Elements data periodically) is implemented.
- larger threshold values can be specified now.
- the Log header line listing all elements are now stored at the beginning of each new Statistics Log file.

# XMPP

- the module is re-implemented.
- multiparty chat (XEP-0045) is implemented. See the Named Tasks section.
- vcard-temp (XEP-0054) is implemented.
- XMPP Ping (XEP-0199) is implemented.
- output channel statistic elements have been implemented.
- the XMPPS and XMPPI System Log tags are swapped (XMPPS is used for client sessions now).
- XEP-0055 (Jabber Search) is implemented.
- support for XEP-0085 (Chat State Notifications) is implemented.

# LDAP

- the "extended match" operation is supported.
- the Router Subtree feature is implemented.

# WebAdmin

- the "Pronto" section has been added to the Account Preference page.
- additional Local Account Manager settings are implemented.
- the Temporarily Blocked Addresses -> Block After setting is documented.

# WebUser

- the Kazakh language has been added.
- the ';' symbol is accepted as a To/Cc/Bcc address list separator.
- the Billing and Mobile Manager pages are implemented.
- the Personal Info (vCard Profile) editor page is implemented.
- the "Classic" skin is retired.
- the Attach Profile vCard message composer option is implemented.
- the RSIP Signal settings page is implemented.
- the Chatroom Log viewer is implemented.
- the Contact Editor now supports the Yomi Company Name and the Yomi Name fields.
- the FireFox "x-dns-prefetch-control" meta-tag is added to disable the spammers DNS-based "harvesting" techniques.
- plain text space formatting is improved ("ASCII-tables" are supported now).
- the Send Requests option is disabled by default if an Event is composed not in the Default Calendar mailbox.

# XIMSS

- the retrieveURL operation is renamed into httpCall (the old name is still available).
- the httpCall "field" elements can contain non-string data now.
- the reqSeq parameter is documented.
- the listUploaded operation is implemented.
- the newUploaded data message is implemented.
- the makeCall function now completes immediately, the Click-to-Call status reports come as asynchronous data messages now.
- the fileAttr[Read|Write], fileSub[List|Update] operations are implemented, the timeAttrModified attribute is added to the fileInfo data messages.
- the file[Read|Write] type attribute with "vcard" value is supported now.
- the iqSend/iqRead operations are implemented.
- the sendIM/readIM attributes are extended.
- the messageAttr[Read|Write] operations are implemented
- the datasetRemove operation can process foreign datasets now.

- when reading messages, the parts with "attachment" disposition are not included into message XML.
- the messageAdded message can be sent in response to messageAppend and messageCopy requests.
- the messageSubmit operation now copies the "Bcc" field data into the saved message.
- */xml and */*+xml message parts are supported now.
- the rsipList operation is implemented.
- the datasetListSubset operation is implemented.
- multi-address E-mail fields are composed as single RFC Header fields.

## CLI

- the GETSNMPELEMENT operation renamed into GETSTATELEMENT, its result format is changed to Number.
- the *WEBFILE* operations are renamed into *STORAGFILE* operations.
- the [READ|UPDATE]STORAGEFILEATTR operations are implemented.
- the GETNEXTSTATNAME, [GET|SET]MEDIASERVERSETTINGS commands are implemented.
- the [GET|SET]ACCOUNTSUBSCRIPTION commands are renamed into [GET|SET]MAILBOXSUBSCRIPTION.
- the [GET|SET]FILESUBSCRIPTION commands are implemented.
- the *NAMEDTASK commands are implemented.
- the [GET|SET]ACCOUNTRPOP commands are removed. The [GET|SET]ACCOUNTRPOPS commands are implemented.
- the [GET|SET]ACCOUNTRSIPS commands are implemented.
- the Directory Administration commands are implemented.
- the SETACCOUNTPASSWORD command supports the optional CHECK keyword now.

## WSSP

- the INCLUDE elements now accepts an expression as a filename.
- the JAVASCRIPT: text element now escapes the < symbols (a workaround for browser bugs).
- the JSON: text element is implemented.

## CG/PL

- the "for-loop" operator is implemented.
- qualified external code names (moduleName::codeName) are implemented.
- the builtin FUNCCALL function and the PROCCALL procedure are implemented.
- the UPDATESTATISTICS procedure is implemented.
- the RADIUSCall function parameters are extended.
- the MAILBOXCOPYBYUID and MAILBOXSETFLAGSBYUID function renamed to COPYMAILBOX and MAILBOXSETFLAGS (old names are preserved), function parameters are extended.
- the [READ|WRITE|APPEND|RENAME|DELETE|LIST]SITE[FILE|DIRECTORY] functions are renamed into [READ|WRITE|APPEND|RENAME|DELETE|LIST]STORAGE[FILE|DIRECTORY].
- the [READ|WRITE]STORAGEFILEATTR functions are implemented.
- the [GET|UPDATE]MESSAGEATTR functions are implemented.
- the LISTSITEFILES function result is changed (the elements are always dictionaries, the MetaModified timestamps are added).
- the SENDINSTANTMESSAGE function is extended, the SENDXMPPIQ function is implemented.
- the READENVIRFILE function is documented.
- the MAILBOX[ORIGUIDBYUID|UIDBYORIGUID] functions are implemented.
- the comparison operations can be applied to strings and datablocks.
- the MAILBOXCOPY function parameters are extended.
- the DATASETLISTSUBSETS function is implemented.

## PBXApp

- the SETLOCALCONTACTPARAMETER procedure is implemented.
- the Alert-Info SETCALLPARAMETERS dictionary element is supported now.
- the SETINTERRUPTONDTMF function now controls DTMF interrupts for both audio playing and audio recording operations.
- Instant Message and XMPP IQ receiving, the ISINSTANTMESSAGEEVENT and ISXMPPIQEVENT functions are implemented.
- the DisconnectEx procedure is implemented.

## SPELLCHECK

- UTF-8 "symbol" offsets are supported.

CommuniGate Pro

**Introduction** | **Features** | **History** | **How To** | **Help Me**

# Version 5.2 Revision History

- **6.2**
- **6.1**
- **6.0**
- **5.4**
- **5.3**
- **5.2: Summary**
- **5.1**
- **5.0**
- **4.x**
- **3.x**
- **2.x**
- **1.x**

- **Pronto!**
- **MAPI Connector**

RSS

**5.2.20** 15-Apr-2010
`Valid Core License Keys:` issued after 01-Jan-2007.

- Bug Fix: CALLLEG: 5.1.2: the Replaces "method" parameter was checked incorrectly.
- Bug Fix: CG/PL: 5.1c1: the DateByYearDate function returned a result for a previous year.
- Bug Fix: SIGNAL: 4.3c1: minimum and maximum REGISTER period limits were not applied to the "expires" Contact parameters.
- Bug Fix: SIGNAL: 5.2c2: the Dialog Info data was not copied when retrieved, causing various problems under heavy load.
- Bug Fix: DIRECTORY: 3.0: Local Unit modification records longer than 32K could not be read.
- Bug Fix: MAILBOX: 5.2c1: hard-links were never used for messages in MailDir mailboxes.
- Bug Fix: Kernel: 5.2.4: when an external certificate was verified, the stock certificates were used incorrectly.
- Bug Fix: CLI: 5.2c1: incorrect parameters for the GETACCOUNTSTAT and RESETACCOUNTSTAT commands could crash the server.
- Bug Fix: WebUser: 5.2.10: the HiddenSkins setting could not be used to hide the "unnamed" (Basic) Skin.

**5.2.19** 07-Jan-2010
`Valid Core License Keys:` issued after 01-Jan-2007.

- Bug Fix: SIGNALRULES: 5.0c1: malformed rule could cause a crash.
- Bug Fix: XMPP: 5.2c2: broken connections to a remote server could cause a crash.
- Bug Fix: HELPERS: 5.1c1: after a server restart, the "never" Filter Helper time-out values could be set to non-zero values.

- Bug Fix: CALENDAR: 5.2c1: converting the recurrence "count" value into "until" value could add an additional day.
- Bug Fix: CALENDAR: 5.2.11: old-style "duration" data parsing could fail.

**5.2.18** 24-Nov-2009
`Valid Core License Keys:` issued after 01-Jan-2007.

- Bug Fix: CLUSTER: 5.2c1: virtual (inter-cluster) Mailbox closing could crash the server.
- Bug Fix: SIP: 5.2.1: some fields could be composed without proper URL encoding.
- Bug Fix: ACCOUNT: 5.1c3: SIP "Event Publish" operation could cause memory corruption.

**5.2.17** 31-Oct-2009
`Valid Core License Keys:` issued after 01-Jan-2007.

- MAILBOX: forward-compatibility with v.5.3x mailbox formats is implemented.
- MEDIA: Inband DTMF detection is redesigned.
- Bug Fix: MAILLIST: 5.2c1: created subscriber records had incorrect format; bounce time updates caused errors.
- Bug Fix: Admin: 5.1: the "account import" function incorrectly processed Account Signal Rules.
- Bug Fix: HTTP: 5.2c3: reading "default" (or "index") directory files could fail if the "partial read" method was used.

**5.2.16** 08-Aug-2009
`Valid Core License Keys:` issued after 01-Jan-2007.

- Bug Fix: DIRECTORY: 5.2c1: Remote Unit Security setting could not be set via WebAdmin.
- Bug Fix: AIRSYNC: 5.1c4: EOL processing has been improved to avoid iPhone event update looping.

**5.2.15** 15-Jul-2009
`Valid Core License Keys:` issued after 01-Jan-2007.

- AIRSYNC: "Yomi" attributes are supported now.
- AIRSYNC: a workaround for iPhone bug causing Calendar event loops is implemented.
- Kernel: UTF-8 converter detects all "bad" sequences now.
- Bug Fix: XMPP: 5.1c5: the 'from' attribute for the "iq" results now contains the full account name (jid).
- Bug Fix: HTTPO: 5.1c5: "Basic" authentication now uses the mixed-case "Basic" keyword.
- Bug Fix: WebUser: 5.1.2: links in plain text messages could be processed incorrectly.
- Bug Fix: AIRSYNC: 5.1c4: All-Day Events in "eastern" time zones were composed incorrectly.
- Bug Fix: Media: 5.2.4: short inband DTMF "fantoms" were incorrectly detected.
- Bug Fix: CLUSTER: 5.2c4: the CREATEMAILBOX command with both CLASS and AUTH clauses did not work.
- Bug Fix: WebAdmin: 5.2c1: domain admins could not update Domain-level Client IP Address settings via WebAdmin.

**5.2.14** 26-Apr-2009
`Valid Core License Keys:` issued after 01-Jan-2007.

- MAPI: the [MAPI Connector version 1.52.14.1](#) is included.
- AIRSYNC: a workaround for the Apple iPhone bug in processing multipart/alternative messages.
- Bug Fix: CG/PL: 5.1c2: the MAILBOXREDIRECTBYUID function was equivalent to the MAILBOXFORWARDBYUID function.
- Bug Fix: CLUSTER: 5.2.10: mailbox-alias name resolving procedure could fail when used with a Cluster

Account.

- Bug Fix: CG/PL: 5.2c1: the SETACCOUNTPREFERENCES operations reset the Language and TimeZone environment variables.
- Bug Fix: XIMSS: 5.2c2: "auto-built" mail attributes may not be built for certain Remote Directory Units.
- Bug Fix: WebUser: 5.0c2: deleting Contact Group elements could cause program exceptions.
- Bug Fix: GROUPS: 4.0: some group cross-references could cause infinite loops.
- Bug Fix: PBX: 5.1c2: the Record operation was started even if there was a pending event.
- Bug Fix: DOMAIN: 5.2.2: renaming Accounts in a suspended Domain could crash the server.

**5.2.13** 08-Mar-2009
`Valid Core License Keys:` issued after 01-Jan-2007.

- MAPI: the [MAPI Connector version 1.52.13.1](#) is included.
- PKI: the Certificate Signing Requests now use the SHA1 algorithm instead of MD5.
- HTTPO: a workaround for Perl form data parser bug is implemented.
- SIGNAL: the *NN addresses are processed in the Redirect/Fork responses and Rule actions.
- CG/PL: the MailboxExpunge operation updates the mailbox view first.
- XIMSS: the prefsReload operation is implemented.
- Kernel: the ms874 and ms-874 charset names are supported now.
- Kernel: when composing RFC822 fields, UTF-8 encoded atoms are composed using the shorter of QP or Binary encoding.
- ROUTER: the .local and .domain suffix processors now replace the '%' symbol with '@' in the routed "local" address part.
- WebUser: when renewing a user S/MIME certificate, the certificate serial number is changed now.
- Bug Fix: CLI: 5.2.5: the CREATEWEBSESSION command incorrectly processed the IMode and IModeJP keywords.
- Bug Fix: SDP: 5.1c1: the document-level "p" attribute was parsed incorrectly.
- Bug Fix: XIMSS: 5.2c1: folder filtering by header fields could work incorrectly.
- Bug Fix: AIRSYNC: 5.2c1: when a Submit operation used Base64 encoding, the last message line could be lost.
- Bug Fix: WebUser: 5.2.12: the call logs page did not convert the call start time into the local time zone.
- Bug Fix: CalDAV: 5.2c3: if multiple invitation recipients were specified in a single HTTP field, only the first one was used.
- Bug Fix: Foundation: 5.2c1: the Use OS Password option did not work on the OS/400 platform.
- Bug Fix: IMAP: 5.2.8: the APPEND command could fail with some foreign mailboxes.
- Bug Fix: Kernel: 5.2.11: the Hide 'Account Unknown' messages option did not work for connection-based protocols (FTP, IMAP, POP, LDAP, etc.)

**5.2.12** 23-Jan-2009
`Valid Core License Keys:` issued after 01-Jan-2007.

- MAPI: the [MAPI Connector version 1.52.12.1](#) is included.
- PKI: the default Certificate signing algorithm has been changed from MD5 to SHA1.
- AIRSYNC: Calendar item updating changed to provide a workaround for the Apple iPhone 2.2 "looped update" bug.
- AIRSYNC: workaround for iPhone bug causing it to crash on some vCards
- CG/PL: the DatasetRemove, DatasetDelete functions are implemented.
- HTTP: the crossdomain.xml file data (needed for Flash XIMSS clients) moved to a Skin file now.
- Foundation: indefinite-length elements in message signatures are supported now.
- HTTP: when a CG/PL application is called using the /sys/ realm, the "sysEntry" code section is used instead of the "main" section.
- Bug Fix: CLUSTER: 5.2.11: HTTP Session requests in a cluster were processed incorrectly.
- Bug Fix: Network: 5.2.11: "Banned on failed Login" function could block White-listed addresses.
- Bug Fix: XIMSS: 5.2c1: vCardGroup XML parser did not process member real names.
- Bug Fix: XIMSS: 5.2c1: some folderRead operations ignored the "partID" attribute.

- Bug Fix: XIMSS: 5.2c1: the "password recovery" operation did not work in a Cluster.
- Bug Fix: CALENDAR: 5.0: Calendar parsing could not process mime-encoded message bodies.

**5.2.11** 19-Dec-2008
`Valid Core License Keys`: issued after 01-Jan-2007.

- Pronto: the [Pronto version 2.7](#) is included.
- MAPI: the [MAPI Connector version 1.52.11.2](#) is included.
- DNR: the response packet size limit is increased to 64K.
- CG/PL: the compiled module code size limit is increased from 64Kb to 1Mb.
- CG/PL: the LISTMAILBOXES, [GET|SET]MAILBOX[SUBSCRIPTION|ALIASES] functions are implemented.
- CG/PL: the MailboxAppend function now supports the "report" element.
- XMPP: the "optional" tag is added to "features" elements.
- ACCOUNT: the Upload File Charset preference is implemented.
- TLS: if a client provides a certificate with a wrong issuer and the Domain Require Certificate option is not enabled, the certificate is ignored without rejecting the client.
- CALDAV: user E-mail address and Real Name is now taken from the Account Preferences.
- SIGNAL: the 503 code is proxied upstream as the 500 code now.
- SIGNAL: the "Failure" stage rules are not applied for Busy or NoAnswer conditions.
- XIMSS: changing message flags in folders filtered by the flags values removes messages from and adds messages to the folder view.
- Bug Fix: Foundation: 4.0: the XML/HTML converter did not encode ASCII control symbols.
- Bug Fix: CALENDAR: 5.2.10: the Accept operation did not change the busy status of ACCEPT'ed Events to Busy, and Outlook-sent events were copied as "tentative".
- Bug Fix: CALENDAR: 4.3: ALERT duration parser did not accept some formats.
- Bug Fix: WebUser: 5.2.5: "safe link mode" created incorrect URLs for links with "anchors".
- Bug Fix: XIMSS: 5.1.8: the folderReopen operation did not keep the existing "view field set" if no field was specified.
- Bug Fix: XIMSS: 5.2c2: the "peer" attribute of the callTransfer operation was not converted into a "full name" URI.
- Bug Fix: CLUSTER: 5.2c2: if the markup mode (IMode, WML, HTML) was specified as a login page URL, it was not passed to the backend servers on login.
- Bug Fix: CLUSTER: 5.2c4: it was not possible to do execute the startPBXTask operation on a remote frontend.
- Bug Fix: NETWORK: addresses used for failed Login operations were not added to the Temporarily Blocked Addresses list.
- Bug Fix: AIRSYNC: 5.2.10: the Email-type header fields (From, To, etc.) did not include the "real name" part.
- Bug Fix: AIRSYNC: 5.2.1: new Event and Task objects were not fully initialized, resulting in garbage data being stored with some clients.
- Bug Fix: AIRSYNC: 5.2.9: if a client removed the Calendar ALARM item, it was not removed on the Server.
- Bug Fix: AIRSYNC: 5.2c2: the Search operation now finds only those Directory records where the mail attribute is present or can be present.
- Bug Fix: SIGNAL: 5.2c1: Instant messages generated within XIMSS/XMPP sessions and with Automated Rules were sent using non-authenticated Signals.
- Bug Fix: CALLLEG: 5.1.7: reliable 180-SDP was not reused when an 200 SDP-less response came for a synchronous call (such as those used in the "MakeCall" feature).
- Bug Fix: CALLLEG: 5.0.2: requests without a to-tag could be accepted, producing (for BYE requests) abandoned DIALOG objects.
- Bug Fix: CG/PL: 5.2.10: message copy operations could crash the server.
- Bug Fix: MAILBOX: 5.2.10: if a TextMailbox contained a message with a "remote past" INTERNALDATE, mailbox parsing could fail.
- Bug Fix: ACCOUNT: 5.2.c3: in some situations "direct" ACLs rights for a particular account could be merged with the "anyone" access rights.
- Bug Fix: BSDLog: 5.1c2: the "Use OS syslog" option could crash the server.

**5.2.10** 05-Nov-2008
`Valid Core License Keys:` issued after 01-Jan-2007.

- Pronto: the Pronto version 2.6 is included.
- CG/PL: the MessagePart function result now can include the filename element.
- CG/PL: the MessageFields function is replaced with the MessageHeader function (the MessageFields function is still supported).
- CG/PL: the MailboxCopyByUID function is implemented.
- WebAdmin: the BannerInfo and HiddenSkins Account settings are implemented.
- CLI: the ROSTER and BILLING commands are documented.
- XIMSS: the fileRead and fileWrite operations now support the type=xml attribute.
- XIMSS: the findEvents operation now supports the byAlarm attribute.
- SMTP: the reverse-connect function now uses a 30-second time-out when establishing a tcp connection in the Non-Blocking socket mode.
- Bug Fix: CALLLEG: 5.2.3: Record-Route sets for 1xx and 2xx responses could be built differently.
- Bug Fix: RULES: 5.2c2: "merged" Server and Account/Domain rules were sorted "linearly" in Signal Rules (i.e. Server rules were not placed in front or after all Account/Domain rules).
- Bug Fix: FTP: 4.0: in some situations connections were not re-initialized before being reused.
- Bug Fix: GSSAPI: 5.0c2: some operations with the "NEGOTIATE" method objects could cause loops or stack overflow.
- Bug Fix: CALENDAR: 4.1: "start of the epoch" dates (01-01-1970) were processed as "incorrect dated".
- Bug Fix: CALENDAR: 5.0: the Status Update operation changed the event own busy status to match the Event Reply status.
- Bug Fix: TLS: 5.0: if a client certificate was ignored for any reason, the CertificateVerify TLS record was interrupting TLS negotiations.
- Bug Fix: AIRSYNC: 5.2c3: meeting updates by an attendee caused protocol errors.
- Bug Fix: AIRSYNC: 5.2c2: messages with the "service" flag were sync'ed with mobile devices.
- Bug Fix: KERNEL: 5.2c1: on some HP/UX systems only local IPv6 addresses were detected.

**5.2.9** 19-Oct-2008
`Valid Core License Keys:` issued after 01-Jan-2007.

- Pronto: the Pronto version 2.5 is included.
- WebUser: the Albanian language has been added.
- AIRSYNC: "MIME" support added (used with iPhone to download attachments).
- AIRSYNC: Calendar/Task "reminders" are supported now.
- SMTP: now the Domain-level "Client IPs" are used to enable message relaying.
- Kernel: SPNEGO authentication now accepts "in-line" mesh tickets.
- LOG: the External Log "Facility Code" option is implemented.
- QUEUERULES: the Store In action now supports the [IFEXISTS] and [MUSTEXIST] prefixes.
- XIMSS: the contactFind operation is implemented.
- ACCOUNT: the HistoryBox preference is implemented.
- AIRSYNC: the Enable/Disable PIN Lock Admin functions have been implemented.
- HTTP: the single MIME Extension table is replaced with the "built-in" and "custom" tables.
- TLS: when checking the Certificate subject, multiple "altSubjectName" elements are checked now.
- DOMAIN: The External AUTH protocol is extended to support the [PRE|POST]TYPECHANGE, [PRE|POST]UPDATE, and the PREPWDCHANGE commands.
- WebAdmin: the Domain Object List includes Mailing List objects now.
- Helpers: the External Banner System API, its XIMSS and CG/PL interfaces are documented.
- PBXApp: the IsPlayCompleted function is implemented.
- IMAP: the "Non-Mail Folders visible via IMAP" Account setting is implemented.
- SIP: CANCEL/200 race condition processing added (BYE request generator implemented).
- CLI: the [GET|SET]SESSIONSETTINGS commands are implemented.

Lawful Intercept: when an Account is renamed/removed, its LI record is updated/removed.

- Bug Fix: WebAdmin: 5.2c2: the active secondary Logs could not be read on some platforms.
- Bug Fix: BSDLog: 5.1c1: timestamp in records sent to remote syslog servers did not include seconds.
- Bug Fix: ACCOUNT: 5.2c2: some account "login info" could be discarded without updating.
- Bug Fix: CLUSTER: 5.2.8: the account aliases in shared Domains could be set on the Controller only.
- Bug Fix: Admin: 5.2.7: when ACL included groups from other Domains, some group member names were processed incorrectly.
- Bug Fix: Kernel: 4.0: the Egypt/South Africa zone in the TimeZones.data file contained an incorrect time offset value.
- Bug Fix: RULES: 5.2c2: "merged" Account and Domain rules were sorted "linearly" in Signal Rules (i.e. Domain rules were not placed in front or after all Account rules).
- Bug Fix: XIMSS: the CLI CREATEXIMSSSESSION operation and some forms of HTTP Login did not update the LastLogin Account info.
- Bug Fix: HTTP: 5.2.6: running CG/PL applications without authentication could crash the server.
- Bug Fix: AIRSYNC: 5.2c3: some date-time fields were composed in the RFC2445 format instead of the ISO format.
- Bug Fix: AIRSYNC: 5.2c2: iCalendar ALERT data was processed incorrectly.
- Bug Fix: AIRSYNC: 5.2.4: client-updating items that have been just added with the same client could produce duplicated items on that client.
- Bug Fix: GROUPWARE: 3.0: line-wrapping could break a multi-byte symbol in the middle.
- Bug Fix: WebAdmin: 5.2c1: the Class of Service could not be set in an Account Template.
- Bug Fix: PBXApp: 5.1c1: when a StartBridgedCall operation was canceled because the "other" task quited or it incoming call was canceled, no event was delivered to the current Task.

**5.2.8** 12-Sep-2008
`Valid Core License Keys`: issued after 01-Jan-2007.

- The Billing subsystem is documented.
- Pronto: the [Pronto version 2.3.1](#) is included.
- CLI: the GETACCOUNTPRESENCE command is implemented.
- WSSP: the STRINGMENU operator is case-sensitive now.
- GROUPWARE: now long base64-encoded VCard properties are stored using the "folded" format.
- QUEUERULES: the 'From' condition processing of messages without any From address is changed.
- Bug Fix: CLUSTER: 5.2.7: PBX task starting failed if the owner Account could not be opened locally.
- Bug Fix: CLUSTER: 5.2.6: some components incorrectly "remembered" the backend server an Account should be opened on.
- Bug Fix: TEMPFILES: 5.1c1: CG/PL, XIMSS, WebUser operations could generate messages with non-unique Message-IDs if the messages were appended to mailboxes and not sent as E-mails.
- Bug Fix: Admin: 5.1c3: it was possible to create an alias with the same name as a Group or a Mailing List.
- Bug Fix: WebAdmin: 5.2c3: Class Of Service settings pages were processed incorrectly.
- Bug Fix: WebUser, Pronto: 5.2c2: Photo objects added to vCard items did not have the image type flag (JPEG, GIF).
- Bug Fix: CG/PL: 5.2.5: AppToXML could crash on VFreeBusy objects.

**5.2.7** 31-Aug-2008
`Valid Core License Keys`: issued after 01-Jan-2007.

- Pronto: the [Pronto version 2.3](#) is included.
- Admin: ACL can now include groups from other Domains.
- WebUser: if an alternative Skin and/or Language was selected on login, the "restored" session will use it, too.
- CLI: the DELETE[SERVER|CLUSTER|DOMAIN]PBX commands are implemented.
- CLI: the [SET|UPDATE][SIGNAL|MAIL][RULE|RULES] commands can be used for self-management.
- CG/PL: the RANGE, Roster management functions are implemented.
- CG/PL: the READSITEFILE, WRITESITEFILE, MAILBOXAPPEND, OPENMAILBOXVIEW functions are

extended.

- CG/PL: the OBJECTTOJSON, JSONTOOBJECT functions are implemented.
- XIMSS: the messageAppend, contactAppend operations now support the checkOld attribute.
- XIMSS: the Roster, Dataset operations can access other Accounts now.
- XIMSS: the folderOpen, folderReopen operations can use the UIDValidity and UIDMin attributes.
- SMTP: the Send Secure to Domains list can contain "negative" domain names, disabling TLS when sending to those domains.
- MAILBOX: virtual (inter-cluster) Mailbox processing has been optimized.
- Media: the Inband DTMF Threshold setting is implemented.
- Account: the virtual "$DomainPBXApp", "$ServerPBXApp", "$ClusterPBXApp" File Storage directories are implemented, the "$CommonSkins" directories are renamed into the "$ServerSkins" and "$ClusterSkins" ones.
- DIALOG: now "Music on Hold" is started only when the dialog is already in the "connected" state.
- AUTH: now the SessionID method can be used for sessions protected with a cookie.
- Bug Fix: AIRSYNC: 5.2c2: the GAL search operation did not retrieve the 'mail' attributes composed automatically.
- Bug Fix: AIRSYNC: 5.2c2: Add/Fetch response elements could have incorrect types.
- Bug Fix: WebAdmin: 5.2.6: PBX language folders could not be displayed.
- Bug Fix: WebUser, XIMSS: 5.1c1: the E-mail Redirect/Forward operation did not MIME-encode the new To header field.
- Bug Fix: WebAdmin, CLI: 3.0: syntax errors in account names could result in incorrect inter-cluster commands.
- Bug Fix: SIP: 5.2.6: NAT traversal Record-Routes used the illegal "_" symbol - changed to 'C'/'S'.
- Bug Fix: SIP: 5.1c3: CANCEL requests for non-existent requests could stay in the SIPS queue forever.
- Bug Fix: Admin: 5.2.2: sending E-mail to a suspended domain could crash the server.
- Bug Fix: WebUser: 5.2c2: when replying to HTML letters, the "Use HTML" composer flag was set.
- Bug Fix: XIMSS: 5.1c2: authentication methods with "final exchange" (such as DIGEST-MD5) did not work in a Cluster.
- Bug Fix: CG/PL: 5.1c2: the current task effective preferences did not change after the SetPreference() operation.

**5.2.6** 28-Jul-2008
`Valid Core License Keys:` issued after 01-Jan-2007.

- Media Plugin: Firefox on Windows, Safari/Firefox on MacOS X are supported now.
- Pronto: the Pronto version 2.2 is included.
- MAPI: the MAPI Connector version 1.52.5.2 is included.
- Admin: "Pronto-xxxx", "Admin-xxxx" Skins are implemented.
- Admin: the HiddenSkins Account setting is implemented.
- Kernel: the WebPronto directory retired, its files are moved to WebSkins directories.
- Kernel: start parameters are recorded in the System Log now.
- IMAP: now the "CHILDREN" LIST extension is automatically enabled.
- XIMSS: the readStrings, skinFileList, skinFileRead operations now have the skinName parameter; the skinList command is added.
- XIMSS: the fileDirList operation is implemented.
- MEDIA: automatically adjusted mixer delay is implemented.
- Account: virtual "$DomainSkins" and "$CommonSkins" File Storage directories are implemented.
- WebUser: the Greek language has been added.
- QUEUERULES: the Accept Request "decline reason" message is customizable now.
- HTTP: Cache-Control headers added to "plain" files.
- CALLLEG: now session-expiration data is processed in mid-dialog responses, too.
- CG/PL: the message composer can include parts of exiting messages into a new one.
- FTP: the "CWD" operation now checks that the target directory exists.
- FTP: the Legacy-Style LIST option is implemented.
- Bug Fix: Foundation: 5.1c1: XML parser could incorrectly parse "CDATA" data.
- Bug Fix: CLUSTER: 5.2.5: the CheckAccountPassword command did not work.
- Bug Fix: AIRSYNC: 5.2.2: a failure to submit an iCalendar invitation could crash the server.

- Bug Fix: CG/PL: 5.2.5: Mailbox operations failed to "normalize" INBOX names.
- Bug Fix: IMAP: 5.2c4: under very heavy CPU load, an error in a "long" IMAP COPY operation could cause server crash.
- Bug Fix: DOMAINS: 5.2c1: settings read from Directory-based Domains could fail because of case-sensitivity.
- Bug Fix: WebAdmin: 5.2c3: updated WebAdmin preferences were not displayed immediately.
- Bug Fix: CLUSTER: 4.2: "cluster remote services" member sets were not properly re-initialized after a controller failover.

**5.2.5** 03-Jul-2008
`Valid Core License Keys:` issued after 01-Jan-2007.

- Pronto: the [Pronto version 2.1.3](#) is included.
- MAPI: the [MAPI Connector version 1.52.5.1](#) is included.
- AIRSYNC: Sync requests w/o a GetChanges command return the current SyncID (a workaround for Sony devices).
- AIRSYNC: FolderSync requests to foreign Mailboxes are supported now.
- AIRSYNC: PING processor now stores the request parameters.
- ROUTER: local object name parsing changed to support "program{param#param}#account" addresses.
- WebUser: the plain-text URL detection algorithm has been changed to allow for URLs containing the "()" and "[]" symbols.
- WebUser: the Directory Address Book search pattern is specified as the "AddressBookSearch" Skin string now.
- Kernel: message charset processing changed to eliminate incorrect UTF-8 sequences.
- CG/PL: the [CREATE|RENAME|DELETE]MAILBOX, [GET|SET]MAILBOXACLS functions are implemented.
- CG/PL: the APPTOXML and XMLTOAPP functions are implemented.
- CG/PL: the MAILBOXAPPEND function now accepts vCard and vCardGroup contents.
- XIMSS: the fileRead, fileWrite operations new accept the "object" data type.
- HTTP: the CLI (Text/SOAP) access is implemented.
- DIALOG: the "x-moh" attribute is added to the Music-on-Hold SDPs to avoid "multiple MOH" effects.
- GROUPWARE: X-Telnum fields now contain "pure" phone numbers.
- GROUPWARE: X-File-As attribute is added to vCard data.
- Bug Fix: CALLLEG: 5.2.4: the "Via" fields were not re-used when an request was resent because of a 401/407/491/etc. response.
- Bug Fix: XMPP: 5.2c3: the XMPP sending WebAdmin Monitor displayed XMPP reading data.
- Bug Fix: WebApp: 5.2c1: CG/PL applications did not load User Preferences.
- Bug Fix: AIRSYNC: 5.2c4: a server could send an "item deleted/updated" stanza before sending the item itself.
- Bug Fix: WebUser: 5.2.4: session-less Directory, Signup, and PasswordRecovery pages did not work.

**5.2.4** 23-Jun-2008
`Valid Core License Keys:` issued after 01-Jan-2007.

- Pronto: the Pronto version 2.1.2 is included.
- MAPI: the MAPI Connector version 1.51.16.5 is included.
- SMTP: if a receiving server announces its message SIZE limit, messages larger than that limit are rejected w/o making a delivery attempt.
- CG/PL: the DatasetList, DatasetSet functions are implemented.
- XIMSS: the datasetDelete, fileInfo operations are implemented.
- ACCOUNT: Shared Private Files (alternative file paths) are implemented.
- ACCOUNT: the Concurrent Calls setting is implemented.
- HTTP: the /html login realm is implemented (to force the HTML mode for WML and IMode devices).
- SIGNAL: additional fields have been added to the type 02 CDR records.
- SIGNAL: the Event Limit setting is implemented.
- SIGNAL: the "Random delta" registration setting is implemented.
- SIP: Record-Routes for far-end NAT UDP clients now include the local socket info.

- SIP: the Event Limit settings are implemented.
- REALTIME: the Event Limit setting is implemented.
- ADMIN: the Password Recovery Account setting is implemented.
- ADMIN: now an Account Import file can contain a Telnums column.
- AIRSYNC: devices sending authentication data as "domain/user@domain" are supported now.
- CLI: the FINDWEBUSERSESSION command has been renamed into FINDACCOUNTSESSION command, new optional parameters have been added.
- Kernel: the cp874 charset is supported now.
- ROUTER: a default record for voicemail starting (*nnn) has been added.
- SECURITY: now TLS client connections check the "alternative subject" Certificate attributes before checking the Certificate subject name.
- XMPP: the type attribute is added to outgoing IM messages.
- AIRSYNC: when a device (such as Apple iPhone) sends a new PING request w/o closing the old one, the server closes the old one.
- Bug Fix: CALLLEG: 5.2c2: transfer operations between internal legs could leave "abandoned" dialogs in account info data.
- Bug Fix: CLI: 5.2.0: the LISTWEBFILES operation incorrectly processed the PATH parameter.
- Bug Fix: CLI: 5.2c2: the RENAMEMAILBOX command incorrectly processed the MAILBOX/MAILBOXES flag.
- Bug Fix: SIGNAL: 5.2.1: the "Busy" and "Failure" Rules could be ignored.
- Bug Fix: CG/PL: 5.2c2: the MessageBody function did not accept a null-value as the part ID.
- Bug Fix: CG/PL: 5.2c3: the MailboxOpen function could fail to open "foreign" mailboxes.
- Bug Fix: SIP: 5.0c3: domain-targeting requests (REGISTER, some SUBSCRIBE) could be processed incorrectly if they had Route fields.
- Bug Fix: WebUser: 5.2.1: the "freebusy.wssp" page did not work.
- Bug Fix: CG/PL: 5.2c1: the ExecuteCLI result data was not properly duplicated.
- Bug Fix: LDAP: 5.0.1: "paged" search requests without sorting did not return the "paging cookie" data.
- Bug Fix: SIP: 5.2c1: processing duplicate requests with "local" Route fields could (very rarely) crash the server.
- Bug Fix: AIRSYNC: 5.2c1: incorrect MoveItem responses caused problems for the Apple iPhone client.
- Bug Fix: WebUser: when I-Mode(JP) clients were served the charset in the HTTP header was set to shift-jis instead of shift_jis.
- Bug Fix: SIP: 5.2c1: internally set "Relay-Via" route fields could be processed incorrectly.
- Bug Fix: WebApp: 5.2c2: the Impersonate() CG/PL operation could cause memory corruption.

**5.2.3** 05-May-2008
Valid Core License Keys: issued after 01-Jan-2007.

- Pronto: the Pronto version 2.1.1 is included.
- MAPI: the MAPI Connector version 1.51.16.4 is included.
- DNR: cached SRV records are "load-balanced" according to their "weight"
- SIGNAL: 500-responses for NOTIFY requests do not cause SUBSCRIBE-dialog termination.
- CLI: [GET|SET|UPDATE]ALLACCOUNTSDEFAULTS commands renamed into [GET|SET|UPDATE]SERVERACCOUNTDEFAULTS (old names continue to work).
- PBX: the RingBack application is now interrupted when there is no other device in the alerting (ringing) state.
- CLUSTER: the processing mode (SIP Farm, Submit, Call Leg Host, HTTP Client) setting value can be set to "Auto" now.
- CLUSTER: the HTTP Client operations are "clusterized" now.
- XIMSS: the contactsImport and calendarImport operations are implemented.
- XIMSS: the Account Management operations are implemented.
- Bug Fix: WebUser: 5.2.2: the Edit Draft operation did not preserve the To/Cc/Bcc fields.
- Bug Fix: DIALOG: 5.2c1: malformed SIP requests could crash the server.
- Bug Fix: MEDIA: 5.2c3: statistics collector could cause random DTMF detection events.
- Bug Fix: SIP: 5.2c1: updating settings on the Workaround WebAdmin page could reset other module settings.
- Bug Fix: XMPP: 5.2c1: updating settings on the Sending WebAdmin page could reset other module settings.

- Bug Fix: RPOP: 5.2c1: updating WebAdmin settings pages incorrectly modified remote account lists.
- Bug Fix: MAILBOX: 5.2.2: access right checks could produce incorrect results.
- Bug Fix: MAILBOX: 5.2c2: checking "negative" access rights could crash the server.
- Bug Fix: ACCOUNT: 5.2c3: the MessagesSent statistics element was not updated.
- Bug Fix: CALLLEG: 5.2c1: INVITE-replaces could cause incorrect "dialog" object processing.

**5.2.2** 17-Apr-2008
`Valid Core License Keys:` issued after 01-Jan-2007.

- Pronto: the Pronto version 2.1 is included.
- LOG: the Time Precision setting is implemented.
- LOG: sending to remote syslog servers is suspended, not stopped when an error is encountered.
- Kernel: session <-> node transaction mechanism has been redesigned.
- Kernel: the --DNRUDPReceiverHighPty command line option is implemented, the --SIPUDPReceiverHighPty option is documented.
- SMTP: when a message recipient is routed to the spamtrap address, the sender IP Address is added to the Temporarily Blocked Addresses list.
- QUEUE: Accounts with disabled "Mail" service cannot submit E-mail messages now.
- ACCOUNT: the Mail Outgoing Size Limit setting is implemented.
- DOMAIN: the suspend/resume functionality is implemented.
- AIRSYNC: the Device Management operations and their WebAdmin Interface are implemented.
- AIRSYNC: the GAL Search operation is implemented.
- AIRSYNC: deleted mailboxes are removed from the devices now.
- MAILBOX: the Visible to Mobile Devices option is implemented.
- RULES: the Preference, FreeBusy conditions are implemented.
- RULES: the SendIM action without the "To:" data is always sent using the MAILER-DAEMON sender address.
- QUEUERULES: the CopyAttachment action is implemented.
- SIGNALRULES: the Request Field condition is implemented.
- CLUSTER: the Call Leg Processing setting has been added to the Cluster Settings.
- CLUSTER: the Telnum manager has been split into the Server-wide and Cluster-wide managers.
- SIGNAL: incoming call delivery mechanism for sessions is redesigned (CALLLEGs are created on the Signal host cluster member).
- CLI: the GETACCOUNTACL, SETACCOUNTACL, GETACCOUNTACLRIGHTS, SUSPENDDOMAIN, RESUMEDOMAIN, LISTSERVERTELNUMS, LISTCLUSTERTELNUMS, LISTDOMAINTELNUMS commands are implemented.
- CLI: the optional FOR parameter has been added to CREATE[XIMS|WEBUSER]SESSION commands.
- CG/PL: the profiling mechanism is implemented.
- WebApp: the ProxiedIPAddress, GetHTTPQuery CG/PL functions are implemented.
- WebAdmin: the Account Status page now displays the active Real-Time Queues and Meetings.
- WebUser: the "proxy" Image Display mode is implemented.
- XIMSS: the `prefsModified` async data message is implemented, the `prefsRead` command accepts the element name sets now.
- XIMSS: the `freeBusyRead` operation is implemented.
- XIMSS: the `datsetList`, `datsetSet` operations are implemented.
- XIMSS: the cliExecute command can produce XML-formatted data.
- SIGNAL: SUBSCRIBE requests sent to local domains are processed using the To field address.
- LIST: the "listserver" manager now rejects automatic messages.
- PBX: the simplified VoiceMail rules now specify the reason in the application parameters.
- PBXLEG: the SetBridgedProvision function can be used to set provisioning responses queue length.
- PBXLEG: all SDP sent within a call now have fixed username and origin addresses.
- PBXLEG: SDP-less initial INVITEs are supported for incoming and outgoing calls.
- Bug Fix: WSSP: 5.2.1: MAILBOXMENU item values were translated using the strings mailbox name dictionary.
- Bug Fix: Router: 5.2.1: new SIP URI parser incorrectly processed the "{" and "}" symbols.

- Bug Fix: Router: 5.1c3: slow DNS operations could crash the server during the Router ENUM setting updates.
- Bug Fix: Account: 5.2c1: the MessagesSent account statistics element was not included into "all statistics" data.
- Bug Fix: CALLEG: 5.2.1: INVITE-replaces were processed incorrectly.
- Bug Fix: CG/PL: 5.2c2: the MESSAGEFIELDS function detected E-mail-type fields only.
- Bug Fix: CLUSTER: 5.2c4: CalDAV HTTP requests were proxied incorrectly.
- Bug Fix: ACCOUNT: 5.2c2: preference update operation on a "fresh" Account could crash the server.
- Bug Fix: PBX: 5.2c1: the "call pickup" routines incorrectly checked dialog activities, causing some pickup operations to fail.
- Bug Fix: AIRSYNC: 5.2c3: some recurrent events created with the MAPI connector could not be read by AirSync.
- Bug Fix: AIRSYNC: 5.2c2: submitted messages in the quoted-printable were loosing their Content-Transfer-Encoding header field.
- Bug Fix: XIMSS: 5.1c3: Calendar requests had an incorrect content-class message header field.
- Bug Fix: CALDAV: 5.2c1: Foreign calendars could not be accessed.
- Bug Fix: MAILBOX: 5.1c2: Access rights granted to anyone@otherdomain did not work.
- Bug Fix: CLI: 5.1c3: the SETDOMAINMAILRULES command did not work (the old SETDOMAINRULES had to be used).

**5.2.1** 19-Mar-2008
`Valid Core License Keys:` issued after 01-Jan-2007.

- Pronto AIR version is included.
- The Japanese Guide is included.
- CLI: the LISTDOMAINOBJECTS command is documented (cookie-based listing of extremely large Domains).
- SIGNAL: the "dialog" package now includes the dialog tags info.
- CLUSTER: now the controller checks if the Heartbeat file has not been taken over.
- PBXLEG: the SetCallParameters procedure is documented.
- SIGNAL: dialog package NOTIFY messages list active dialogs first.
- DIALOG: call provisioning responses change the dialog package state now.
- CALLLEG: INVITE/replaces are supported for early dialogs now.
- WebAdmin: the DIALOG and MEDIA monitor pages are implemented.
- WebAdmin: Domain Security Private Keys in the PKCS#8 format can be imported now.
- Account: the Mail Quota settings can be disabled by setting their values to 101%
- Account: the Call Info setting is implemented.
- Account: the Message Size Limit and File Size Limit settings are implemented.
- AIRSYNC: the seen/unseen message flags is synchronized in both directions now.
- Platform: Linux start-up scripts have been modified to better support SuSE-based distributions.
- RADIUS: trailing zero in the username attribute is ignored now.
- DIRECTORY: "record limit" is now applied to cross-unit searches, too.
- XIMSS: the rosterGroupSet operation is implemented.
- XIMSS: the VALARM elements are parsed now.
- PBX: Call Divert (redirection) number can be entered using the *70 function.
- WebAdmin: when switching Domain IP Addresses from some mode to Manual, the current IP addresses are pre-set.
- WSSP: the MAILBOXMENU structure now uses UTF8-encoded mailbox names as menu values.
- Kernel: "pure Unicode" files are detected by leading zeros.
- GROUPWARE: the vCard parser now accepts some incorrectly formatted objects generated with Apple products.
- SIP: the SIP URI parser has been updated to support non-trivial "username" parts.
- SIP: the Use Short Field Names option is implemented.
- SIP: the REGISTER operation now takes into account the Path fields when finding "old" contacts.
- Bug Fix: ParlayX: 5.2c3: multiple bugs are fixed.
- Bug Fix: AirSync: 5.2c3: some data fields could be not saved when updating Contact items.
- Bug Fix: AirSync: 5.2c2: some subfolders could not be synchronized.

- Bug Fix: XMPP: 5.2.0: "remote XMPP host" descriptors were not released, causing memory leaks.
- Bug Fix: IMAP: 5.1.3: in some mailbox multi-access scenarios "EXPUNGE" responses could be lost.
- Bug Fix: SMTP: 5.1c2: under heavy load, the server could crash while executing the shutdown sequence.
- Bug Fix: XIMSS: 5.2c1: XML -> vCardGroup converter could cause memory corruption.
- Bug Fix: XMPP: 5.2c3: input channels were re-initialized incorrectly, causing sporadic login failures.
- Bug Fix: SIGNAL: 5.2c1: CDR type-1 records were generated only when a DIALOG object was present.
- Bug Fix: CALLLEG: 5.1c2: failed incoming calls could result in "stuck" DIALOG objects.

**5.2** 31-Jan-2008
`Valid Core License Keys:` issued after 01-Jan-2007.

- Pronto: the Pronto version 1.5 is included.
- MAPI: the MAPI Connector version 1.51.15 is included.
- CALENDAR: non-standard VALERT iCalendar elements are stored and restored now.
- SIP: when sending responses, only the "Via" port number is used to compose an "alternative" response destination.
- PBX: the "makeCall" (Click-to-call) function now uses the ClickToCall@domain.invalid From address.
- SIGNAL: improved processing for race conditions on multiple 200-INVITE responses.
- SIGNAL: RFC4235 (dialog package) is implemented.
- XIMSS: the startCallInfo operation and callInfo data message are implemented.
- XMPP: Gtalk-compatible iq-delivery is implemented.
- Utils: mailbox renaming options have been added to the MoveIMAPMail utility.
- CLUSTER: Subject Prefix is preserved during inter-cluster message delivery.
- Bug Fix: AIRSYNC: 52c4: items "changed" on the mobile device could disappear from the mobile device after being updated on the server.
- Bug Fix: CLUSTER: 5.2c3: POP3 inter-cluster logins did not work.
- Bug Fix: XIMSS: 5.2c2: the INTERNALDATE and date-type fields retrieved with the folderBrowse/folderSync operations used an invalid XML syntax.
- Bug Fix: WebUser: 5.2c2: a modified Contact "File As" field was not updated in the VCard body.
- Bug Fix: QUEUE: 5.2c1: messages submitted asynchronously could be rejected when a Plugin scanning limit was exceeded.
- Bug Fix: ROSTER: 5.2c2: creating new Roster groups could damage Roster data.

---

**5.2c4** 25-Dec-2007
`Valid Core License Keys:` issued after 01-Dec-2006.

- CALENDAR: iCalendar parsing changed to support post-defined VTIMEZONEs.
- WebAdmin: the Telnum List page is implemented.
- Platform: Linux /sbin/chkconfig is used to create symlinks for startup scripts.
- WebAdmin: secondary logs browser is implemented.
- WebAdmin: the Queue Message browser can show Message contents now.
- Admin: the CanDeleteLogs and CanViewMessages Monitor Access rights are implemented.
- DOMAIN: the ExternalOnProvision setting is implemented. The External AUTH protocol is extended.
- SMTP: reverse-check mechanism checks for loop-back DNS records now.
- AIRSYNC: added items are remembered between sessions now.
- AIRSYNC: multiple mailboxes can be PING'ed now ("push E-mail" for non-INBOX mailboxes).
- WebApp: the GetHTTPField CG/PL function is implemented.
- CLUSTER: object management connections are closed on I/O error only.
- CLI: [GET|SET][QUEUE|SIGNAL|CLUSTER]SETTINGS, CREATEXIMSSSESSION, GETDIALOGINFO commands are implemented.
- CLI: KILLWEBUSERSESSION, GETWEBUSERSESSION commands are renamed into KILLSESSION, GETSESSION.
- WebApp: several CG/PL functions have been added.

- SNMP: the Trap Protocol version setting is implemented.
- DNR: now "Dummy IP" addresses are detected in SRV/MX request responses, too.
- IMAP: the Send 'Running' option is implemented.
- SIP: the TOS Tag option moved from the Receiving to the Sending WebAdmin page.
- WebUser: the Finnish language has been added.
- Bug Fix: WebUser: 5.2c1: managing mailing list subscription could cause "recursive lock" exceptions.
- Bug Fix: SIGNAL: 5.2c1: lack of synchronization in updating a call dialog could cause server memory corruption.
- Bug Fix: CG/PL: 5.2c1: ExecuteCLI command could crash the server.
- Bug Fix: DIRECTORY: 5.2c2: updating the LDAP Unit settings could crash the server.
- Bug Fix: LOGS: 5.2c1: Log Manager settings were not restored after server restart.
- Bug Fix: XIMSS: 5.2c2: the contactAppend operation did not accept a vCardGroup element as its parameter.
- Bug Fix: XIMSS: 5.1.2: the mailbox ACL operations incorrectly processed non-Latin mailbox names.
- Bug Fix: WebUser: 5.2c2: the mailbox setting pages incorrectly processed non-Latin mailbox names.

**5.2c3** 28-Nov-2007
`Valid Core License Keys:` issued after 01-Nov-2006.

- CalDAV: the basic CalDAV (RFC4791) functionality implemented.
- PKI: RFC3279 implemented ("NULL" Algorithm Identifier parameter processing).
- DNR: caching settings are implemented.
- CG/PL: the XML functions are implemented.
- CG/PL: the SortString procedure is implemented.
- RULES: additional macro-symbols are implemented.
- XIMSS: the sessionID authentication is implemented for the "ximsslogin" realm.
- WebAdmin: Last Failed login information is displayed on the Account settings page. The Clear Failed Logins operation is implemented.
- WebUser, XIMSS: message/rfc822-type attachments are sent without encoding now.
- ParlayX: the Call Notification, Payment, and Account Management interfaces are implemented.
- FingerNotifier: non-blocking connection opening with a short (3 seconds) time-out is used.
- Bug Fix: WebAdmin: 5.2c2: LAN IP list page could modify the Client IP List data instead.
- Bug Fix: WebAdmin: 5.2c2: SIP Listener settings could be reset to default when updating the Gateways and Workaround pages.
- Bug Fix: WebAdmin: 5.2c2: the XMPP Sending Monitor page could crash the server.
- Bug Fix: XIMSS: 5.2c2: X-Telnum header fields were parsed incorrectly.
- Bug Fix: MAILBOX: 5.2c2: checking parent mailbox access rights could result in processing loops.
- Bug Fix: MAILBOX: 5.2c1: several components opened mailboxes hiding "hidden" items even from the mailbox owner.
- Bug Fix: CG/PL: 5.2c2: missing SubmitEMail parameters could crash the server.
- Bug Fix: WebMail: 5.2c2: mailboxes with non-Latin names could not be renamed or removed.

**5.2c2** 13-Nov-2007
`Valid Core License Keys:` issued after 01-Nov-2006.

- MAPI: the MAPI Connector version 1.51.14 is included.
- XMPP: the server-to-server protocol implemented. XMPP settings moved to the Real-Time section.
- SIGNAL: automatic protocol (SIP/XMPP) selection is implemented.
- DNS: SRV-operation caching is implemented.
- Admin: the Denied IP Addresses setting is implemented.
- KERNEL: the --SIPUDPSendBuffer, --SIPUDPReceiveBuffer, --DNRUDPSendBuffer, and --DNRUDPReceiveBuffer command line options are documented.
- Admin: the CanControlCalls Domain Admin right is implemented.
- Admin: the HTTP Service setting is implemented.
- Admin: the CallLogs Account setting is implemented.

- Admin: .settings files uploaded to a PBX environment are checked to be valid dictionaries.
- QUEUERULES: the Vacation Rule can have the "EndDate" setting.
- LOCAL: recipient addresses are preserved during inter-cluster delivery.
- IMAP: RFC4466 (extensions), RFC4731 (ESEARCH), RFC4959 (SASL Initial Client Response) are implemented.
- CG/PL: the "method" syntax for function and procedure calls is implemented.
- CG/PL: the binary assignment operators ( +=, -=, etc.) are implemented.
- CG/PL: the spawn expression now can contain a Task startup parameter.
- CG/PL: the COPY, ADDELEMENT, THISTASK functions are implemented.
- CG/PL: the SubmitEMail, OpenMailboxView, SubscribeEvents, Sync, IsMailboxNotifyEvent, MailboxAppend functions are implemented.
- CG/PL: Message Handle functions are implemented.
- CLI: [GET|SET][CLUSTER][NATEDIPS|DEBUGIPS|DENIEDIPS] commands are implemented.
- CLI: the CREATEMAILBOX command can have the CLASS parameter.
- CLI: the FindForwarders command is implemented.
- MAILBOX: Junk mailbox and its Preference settings are implemented.
- MAILBOX: Special Names ($Sent$, $Drafts$, etc.) are implemented.
- CALENDAR: recurrence "until" processing has been improved.
- QUEUERULES: simplified Junk Mail rules are implemented.
- XIMSS: the retrieveURL operation is implemented.
- SIP: RFC4244 implemented.
- HTTPOutput: automatic redirection is implemented for GET requests.
- WebAdmin: Account rename operation now can move Accounts between Domains.
- WebUser: vCard PHOTO elements are displayed inline and can be modified now.
- WebUser: Call Control settings pages now can upload and remove custom Music-on-Hold and RingBack files.
- Bug Fix: XMPP: 5.2c1: certain server-to-server XMPP requests could crash the Server.
- Bug Fix: SIP: 5.2c1: composing some SIP packets could loop exhausting available memory.
- Bug Fix: CLUSTER: 5.2c1: Frontend IP address settings were ignored, backend IP addresses were used instead.
- Bug Fix: CLUSTER: 5.2c1: message delivery on backends could fail.
- Bug Fix: Foundation: 5.2c1: XML parser incorrectly processed the "encoding" attribute for non-UTF-8 charsets.
- Bug Fix: WebAdmin: 5.2c1: mailing list management operations failed.
- Bug Fix: Admin: 5.2c1: the Class of Service operations were implemented incorrectly.

**5.2c1** 23-Oct-2007
```
Valid Core License Keys:
```
 issued after 01-Oct-2006.

- ROUTER: all records are "NORELAY" by default now. PLEASE CHECK YOUR EXISTING Router Records!.
- MAILBOX: the Synchronization snapshots are implemented.
- AIRSYNC: the Windows Mobile Email/Calendaring/Contacts "ActiveSync" synchronization protocol is implemented.
- XIMSS: HTTP binding is implemented.
- Parlay: the Parlay X "Third Party Call Control" and "Call Handling" interfaces are implemented.
- WebApp: CG/PL Web Applications are supported now.
- MEDIA: inband DTMF detection is implemented.
- MEDIA: "play position" operations are implemented.
- Admin: the Account Type Licensing concept is implemented.
- Admin: the Class of Service mechanism is implemented for Account Settings.
- Admin: the AirSync Service setting is implemented.
- Admin: the Temp Blocked Addresses setting is moved from the SMTP Settings to Network->Blacklisted Settings.
- Admin: the Alt RADIUS Password and Alt SIP Password Account settings become standard ones.
- Admin: the Client IP Addresses Domain-level settings are implemented.
- Admin: the CanControlCluster Monitor admin right is implemented.
- Foundation: now XML parser supports complex DOCTYPEs, non-UTF-8 charsets, and processes ENTITY data.

- Foundation: the SHA256 digest algorithm is implemented.
- XIMSS, XMPP: message delivery failure is reported back to the client now.
- SNMP: MIB elements renamed (they start with lowercase letters) to confirm to SNMP naming recommendations.
- LIST: feed-mode header/trailers are not inserted into vCard and iCalendar text messages.
- SIGNAL: Dialog objects are implemented.
- SIGNAL: new "Call" CDR records can be generated now.
- SIGNAL: call logs in the Account File Storage now include the call alerting time, call duration, and call termination reason.
- SIGNAL: the HoldMusic preference is implemented.
- SIGNAL: external Event interface is implemented.
- PBX: incoming call pick up (intercept) is implemented.
- RULES: the Current Date condition can be used to compare dates only (without time).
- QUEUERULES: the Human Generated condition now checks the "automated" header fields added with the AddHeader operations.
- QUEUERULES: the Accept Request action parameter can specify non-default Calendar mailbox. This action can be used in Server-wide Rules now.
- SIGNALRULES: RequestURI Rule condition is implemented.
- SIGNALRULES: the "Ringback" simplified Rule is implemented.
- TLS: the Client Certificate "Required" option is implemented.
- ACCOUNT: the Account-level ACLs (Access Control Lists) are implemented: call control, delegation, mailbox creation.
- ACCOUNT: the MessagesSent, CallsSent, CallsReceived, and Logins statistical elements are added.
- CG/PL: the Min, Max, ExecuteCLI, Base64Encode, Base64Decode, SetApplicationStatus, SetCaseSensitive, RouteENUM procedures are implemented.
- CG/PL: now the Queue and Meeting operations can work with other Accounts.
- PBXLeg: now all SDP generated for a leg have the same SessionID.
- PBXLeg: B2BUA early media (ringback) relaying is supported in the "connected" mode.
- PBXLeg: the AttachMixer function can be used in the disconnected state.
- PBXApp: the SetProvisionCode, SendCallOptions, PendingRequestExData, SetPlayPosition, GetPlayPosition functions are implemented.
- CALLLEG: the StartCall function now supports the P-CGP-Local parameter.
- NODE: Lawful Intercept is processed for impersonating and Signal operations.
- XIMSS: the fileDirCreate,fileDirRename,fileDirRemove, skinFileRead, skinFileList operations are documented.
- XIMSS: the S/MIME functions are implemented.
- XIMSS: the taskStart function is implemented.
- XIMSS: now the alert service messages have gmtTime and localTime attributes.
- XIMSS: vCardGroup support is implemented.
- XIMSS: publicInfo functions are implemented.
- WSSP: the LENGTH prefix is implemented.
- AUTH: the "WEBUSER" authentication method has been renamed into the "SESSIONID" method.
- EXTAUTH: the "READPLAIN" command has been implemented.
- CLI: the *WEBUSER* commands are renamed into *PREFS* commands. The UPDATEACCOUNTPREFS command is documented.
- CLI: the SETACCOUNTTYPE, GETMESSAGEQUEUEINFO commands are implemented.
- SIP: new mechanisms have been added to detect non-trivial packet loops.
- WebAdmin: the Refresh Self-Signed button has been added to the TLS and S/MIME Domain Admin pages.
- KERNEL: the --CanUseSystemPorts Command Line Option is implemented.
- HTTP: large Personal File Storage files are now downloaded in chunks, the download file size limit is removed.
- SMTP: additional statistics elements have been implemented.

# Summary

## Foundation

- The Japanese Guide is included.
- now XML parser supports complex DOCTYPEs, non-UTF-8 charsets, and processes ENTITY data.
- the SHA256 digest algorithm is implemented.

## Kernel

- session <-> node transaction mechanism has been redesigned.
- the --CanUseSystemPorts command line option is implemented.
- the --SIPUDPSendBuffer, --SIPUDPReceiveBuffer, --DNRUDPSendBuffer, and --DNRUDPReceiveBuffer command line options are documented.
- the --DNRUDPReceiverHighPty command line option is implemented, the --SIPUDPReceiverHighPty option is documented.
- Lawful Intercept is processed for impersonating and Signal operations.
- "pure Unicode" files are detected by leading zeros.
- the cp874, ms874, and ms-874 charset names are supported now.
- message charset processing changed to eliminate incorrect UTF-8 sequences.
- the WebPronto directory retired; its files are moved to WebSkins folders.
- start parameters are recorded in the System Log now.
- the SPNEGO authentication now accepts "in-line" mesh tickets.
- the External Banner System API, its XIMSS and CG/PL interfaces are documented.
- when composing RFC822 fields, UTF-8 encoded atoms are composed using the shorter of QP or Binary encoding.

## Platform

- Linux /sbin/chkconfig is used to create symlinks for startup scripts.
- Linux start-up scripts have been modified to better support SuSE-based distributions.

## Pronto

- the Pronto AIR version is included.
- Pronto: the Pronto version 2.7 is included.

## MAPI

- the MAPI Connector version 1.52.13.1 is included.

## AirSync

- the "ActiveSync" Email/Calendaring/Contacts synchronization protocol is implemented.

## CalDAV

- the basic CalDAV (RFC4791) functionality implemented.

## Parlay X

- the "Third Party Call Control" and "Call Handling" interfaces are implemented.
- the "Call Notification", "Payment", and "Account Management" interfaces are implemented.

## Admin

- the Account Type Licensing concept is implemented.

- the Class of Service mechanism is implemented for Account Settings.
- the AirSync Service setting is implemented.
- the Temp Blocked Addresses setting is moved from the SMTP Settings to Network->Blacklisted Settings.
- the Alt RADIUS Password and Alt SIP Password Account settings become standard ones.
- the Client IP Addresses Domain-level settings are implemented.
- the CanControlCluster Monitor admin right is implemented.
- the Denied IP Addresses setting is implemented.
- the CanControlCalls Domain Admin right is implemented.
- the HTTP Service setting is implemented.
- the CallLogs Account setting is implemented.
- .settings files uploaded to a PBX environment are checked to be valid dictionaries.
- the CanDeleteLogs and CanViewMessages Monitor Access rights are implemented.
- the Password Recovery Account setting is implemented.
- now an Account Import file can contain a Telnums column.
- "Pronto-xxxx", "Admin-xxxx" Skins are implemented.
- the HiddenSkins Account setting is implemented.
- ACL can now include groups from other Domains.
- the Enable/Disable PIN Lock AIRSYNC Admin functions are implemented.

## LOG

- the Time Precision setting is implemented.
- the External Log "Facility Code" option is implemented.

## RULES

- additional macro-symbols are implemented.
- the Current Date condition can be used to compare dates only (without time).
- the Preference, FreeBusy conditions are implemented.

## DNR

- the caching settings are implemented.
- SRV-operation caching is implemented.
- now "Dummy IP" addresses are detected in SRV/MX request responses, too.
- the response packet size limit is increased to 64K.

## CLUSTER

- object management connections are closed on I/O error only.
- Subject Prefix is preserved during inter-cluster message delivery.
- now the controller checks if the Heartbeat file has not been taken over.
- the Call Leg Processing setting has been added to the Cluster Settings.
- the Telnum manager has been split into the Server-wide and Cluster-wide managers.
- the processing mode (SIP Farm, Submit, Call Leg Host, HTTP Client) setting value can be set to "Auto" now.
- the HTTP Client operations are "clusterized" now.

## ROUTER

- all records are "NORELAY" by default now. PLEASE CHECK YOUR EXISTING Router Records!.
- a default record for voicemail starting (*nnn) has been added.

## Security

- the "WEBUSER" authentication method has been renamed into the "SESSIONID" method.

- RFC3279 implemented ("NULL" Algorithm Identifier parameter processing).
- the "READPLAIN" command has been implemented.
- TLS client connections check the "alternative subject" Certificate attributes before checking the Certificate subject name.
- the default Certificate signing algorithm has been changed from MD5 to SHA1.
- the Certificate Signing Requests now use the SHA1 algorithm instead of MD5.

## TLS

- the Client Certificate "Required" option is implemented.
- if a client provides a certificate with a wrong issuer and the Domain Require Certificate option is not enabled, the certificate is ignored without rejecting the client.

## Domains/Accounts

- the suspend/resume Domain functionality is implemented.
- the Account-level ACLs (Access Control Lists) are implemented: call control, delegation, mailbox creation.
- the MessagesSent, CallsSent, CallsReceived, and Logins statistical elements are added.
- the ExternalOnProvision setting is implemented. The External AUTH protocol is extended.
- the Mail Quota settings can be disabled by setting their values to 101%
- the Call Info setting is implemented.
- the Mail Outgoing Size Limit, Message Size Limit, and File Size Limit settings are implemented.
- Accounts with disabled "Mail" service cannot submit E-mail messages now.
- Shared Private Files (alternative file paths) are implemented.
- the Concurrent Calls setting is implemented.
- virtual "$DomainSkins" and "$CommonSkins" File Storage directories are implemented.
- when an Account is renamed/removed, its Lawful Intercept record is updated/removed.
- the HistoryBox preference is implemented.
- the External AUTH protocol is extended to support the [PRE|POST]TYPECHANGE, [PRE|POST]UPDATE, and the PREPWDCHANGE commands.
- the Upload File Charset preference is implemented.

## Mailbox

- the Synchronization snapshots are implemented.
- Junk mailbox and its Preference settings are implemented.
- Special Names ($Sent$, $Drafts$, etc.) are implemented.

## QUEUE RULES

- the Vacation Rule can have the "EndDate" setting.
- simplified Junk Mail rules are implemented.
- the Human Generated condition now checks the "automated" header fields added with the AddHeader operations.
- the Accept Request action parameter can specify non-default Calendar mailbox. This action can be used in Server-wide Rules now.
- the CopyAttachment action is implemented.
- the Accept Request "decline reason" message is customizable now.
- the Store In action now supports the [IFEXISTS] and [MUSTEXIST] prefixes.

## PBX

- incoming call pick up (intercept) is implemented.
- the "makeCall" (Click-to-call) function now uses the ClickToCall@domain.invalid From address.
- Call Divert (redirection) number can be entered using the *70 function.

- the Ringback application is now interrupted when there is no other device in the alerting (ringing) state.

## CG/PL

- the profiling mechanism is implemented.
- the Min, Max, ExecuteCLI, Base64Encode, Base64Decode, SetApplicationStatus, SetCaseSensitive, RouteENUM procedures are implemented.
- now the Queue and Meeting operations can work with other Accounts.
- the "method" syntax for function and procedure calls is implemented.
- the binary assignment operators ( +=, -=, etc.) are implemented.
- the spawn expression now can contain a Task startup parameter.
- the COPY, ADDELEMENT, THISTASK functions are implemented.
- the SubmitEMail, OpenMailboxView, SubscribeEvents, Sync, IsMailboxNotifyEvent, MailboxAppend functions are implemented.
- the Message Handle functions are implemented.
- the XML functions are implemented.
- the SortString procedure is implemented.
- the DatasetList, DatasetSet, DatasetRemove, DatasetDelete functions are implemented.
- the [CREATE|RENAME|DELETE]MAILBOX, [GET|SET]MAILBOXACLS functions are implemented.
- the message composer can include parts of exiting messages into a new one.
- the compiled module code size limit is increased from 64Kb to 1Mb.
- the LISTMAILBOXES, [GET|SET]MAILBOX[SUBSCRIPTION|ALIASES] functions are implemented.
- the MAILBOXAPPEND function now supports the "report" element.
- the MESSAGEPART function result now can include the filename element.
- the MESSAGEFIELDS function is replaced with the MESSAGEHEADER function (the MESSAGEFIELDS function is still supported).
- the MAILBOXCOPYBYUID function is implemented.
- the MAILBOXEXPUNGE operation updates the mailbox view first.

## CALLLEG

- the StartCall function now supports the P-CGP-Local parameter.
- INVITE/replaces are supported for early dialogs now.
- the Event Limit setting is implemented.
- now session-expiration data is processed in mid-dialog responses, too.

## PBXLEG

- now all SDP generated for a leg have the same SessionID.
- B2BUA early media (ringback) relaying is supported in the "connected" mode.
- the AttachMixer function can be used in the disconnected state.
- the SetProvisionCode, SendCallOptions, PendingRequestExData, SetPlayPosition, GetPlayPosition functions are implemented.
- the SetCallParameters procedure is documented.
- the SetBridgedProvision function can be used to set provisioning responses queue length.
- all SDP sent within a call now have fixed username and origin addresses.
- SDP-less initial INVITEs are supported for incoming and outgoing calls.
- the IsPlayCompleted function is implemented.

## SIGNAL

- Dialog objects are implemented.
- new "Call" CDR records can be generated now.
- automatic protocol (SIP/XMPP) selection is implemented.
- call logs in the Account File Storage now include the call alerting time, call duration, and call termination

reason.

- the HoldMusic preference is implemented.
- external Event interface is implemented.
- improved processing for race conditions on multiple 200-INVITE responses.
- RFC4235 (dialog package) is implemented.
- SUBSCRIBE requests sent to local domains are processed using the To field address.
- additional fields have been added to the type 02 CDR records.
- the Event Limit setting is implemented.
- the "Random delta" registration setting is implemented.
- 500-responses for NOTIFY requests do not cause SUBSCRIBE-dialog termination.
- the 503 code is proxied upstream as the 500 code now.
- the "Failure" stage rules are not applied for Busy or NoAnswer conditions.
- the *NN addresses are processed in the Redirect/Fork responses and Rule actions.

## SIGNAL RULES

- the RequestURI Rule condition is implemented.
- the "Ringback" simplified Rule is implemented.
- the Request Field condition is implemented.

## MediaServer

- inband DTMF detection is implemented.
- "play position" operations are implemented.
- automatically adjusted mixer delay is implemented.

## CALENDAR

- recurrence "until" processing has been improved.
- non-standard VALERT iCalendar elements are stored and restored now.
- iCalendar parsing changed to support post-defined VTIMEZONEs.

## CONTACTS

- the vCard parser now accepts some incorrectly formatted objects generated with Apple products.
- X-Telnum fields now contain "pure" phone numbers.

## SIP

- the TOS Tag option moved from the Receiving to the Sending WebAdmin page.
- RFC4244 implemented.
- when sending responses, only the "Via" port number is used to compose an "alternative" response destination.
- new mechanisms have been added to detect non-trivial packet loops.
- SIP URI parser has been updated to support non-trivial "username" parts.
- the Use Short Field Names option is implemented.
- the REGISTER operation now takes into account the Path fields when finding "old" contacts.
- Record-Routes for far-end NAT UDP clients now include the local socket info.
- the Event Limit settings are implemented.
- CANCEL/200 race condition processing added (BYE request generator implemented).

## HTTP

- large Personal File Storage files are now downloaded in chunks, the download file size limit is removed.
- the /html login realm is implemented (to force the HTML mode for WML and IMode devices).
- the CLI (Text/SOAP) access is implemented.

- Cache-Control headers added to "plain" files.
- the single MIME Extension table is replaced with the "built-in" and "custom" tables.

## HTTPOutput

- automatic redirection is implemented.
- a workaround for Perl form data parser bug is implemented.

## IMAP

- the Send 'Running' option is implemented.
- RFC4466 (extensions), RFC4731 (ESEARCH), RFC4959 (SASL Initial Client Response) are implemented.
- now the "CHILDREN" LIST extension is automatically enabled.
- the "Non-Mail Folders visible via IMAP" Account setting is implemented.

## FTP

- the "CWD" operation now checks that the target directory exists.
- the Legacy-Style LIST option is implemented.

## SMTP

- reverse-check mechanism checks for loop-back DNS records now.
- additional statistics elements have been implemented.
- if the receiving server announces its message SIZE limit, messages larger than that limit are rejected w/o making a delivery attempt.
- now the Domain-level "Client IPs" are used to enable message relaying.
- the reverse-connect function now uses a 30-second time-out when establishing a tcp connection in the Non-Blocking socket mode.

## LOCAL

- recipient addresses are preserved during inter-cluster delivery.

## SNMP

- the Trap Protocol version setting is implemented.
- MIB elements renamed (they start with lowercase letters) to confirm to SNMP naming recommendations.

## XMPP

- the server-to-server protocol implemented. XMPP settings moved to the Real-Time section.
- instant message delivery failure is reported back to the client now.
- Gtalk-compatible iq-delivery is implemented.
- the type attribute is added to outgoing IM messages.
- the "optional" tag is added to "features" elements.

## XIMSS

- HTTP binding is implemented.
- the fileDirCreate,fileDirRename,fileDirRemove, skinFileRead, skinFileList operations are documented.
- the S/MIME functions are implemented.
- the taskStart function is implemented.
- now the alert service messages have gmtTime and localTime attributes.
- vCardGroup support is implemented.

publicInfo functions are implemented.

- the retrieveURL operation is implemented.
- the sessionID authentication is implemented for the "ximsslogin" realm.
- message/rfc822-type attachments are sent without encoding now.
- instant message delivery failure is reported back to the client now.
- the startCallInfo operation and callInfo data message are implemented.
- the rosterGroupSet operation is implemented.
- the VALARM elements are parsed now.
- the `prefsModified` async data message is implemented, the `prefsRead` command accepts the element name sets now.
- the `freeBusyRead` operation is implemented.
- the `datsetList, datsetSet` operations are implemented.
- the cliExecute command can produce XML-formatted data.
- the contactsImport and calendarImport operations are implemented.
- the Account Management operations are implemented.
- the datasetDelete, fileInfo, fileDirList operations are implemented.
- the fileRead, fileWrite operations new accept the "object" data type.
- the readStrings, skinFileList, skinFileRead operations now have the skinName parameter; the skinList command is added.
- the contactFind operation is implemented.
- the fileRead and fileWrite operations now support the type=xml attribute.
- the findEvents operation now supports the byAlarm attribute.
- the prefsReload operation is implemented.
- changing message flags in folders filtered by the flags values removes messages from and adds messages to the folder view.

## Mailing Lists

- feed-mode header/trailers are not inserted into vCard and iCalendar text messages.
- the "listserver" manager now rejects automatic messages.

## CLI

- [GET|SET][CLUSTER][NATEDIPS|DEBUGIPS|DENIEDIPS] commands are implemented.
- the CREATEMAILBOX command can have the CLASS parameter.
- the FindForwarders command is implemented.
- the *WEBUSER* commands are renamed into *PREFS* commands. The UPDATEACCOUNTPREFS command is documented.
- the SETACCOUNTTYPE, GETMESSAGEQUEUEINFO commands are implemented.
- [GET|SET][QUEUE|SIGNAL|CLUSTER]SETTINGS, CREATEXIMSSSESSION, GETDIALOGINFO commands are implemented.
- KILLWEBUSERSESSION, GETWEBUSERSESSION commands are renamed into KILLSESSION, GETSESSION.
- the GETACCOUNTACL, SETACCOUNTACL, GETACCOUNTFUNCRIGHTS commands are implemented.
- the SUSPENDDOMAIN, RESUMEDOMAIN, LISTSERVERTELNUMS, LISTCLUSTERTELNUMS, LISTDOMAINTELNUMS commands are implemented.
- the LISTDOMAINOBJECTS command is documented (cookie-based listing of extremely large Domains).
- the optional FOR parameter has been added to CREATE[XIMS|WEBUSER]SESSION commands.
- [GET|SET|UPDATE]ALLACCOUNTSDEFAULTS commands renamed into [GET|SET|UPDATE]SERVERACCOUNTDEFAULTS (old names continue to work).
- the FINDWEBUSERSESSION command has been renamed into FINDACCOUNTSESSION command, new optional parameters have been added.
- the DELETE[SERVER|CLUSTER|DOMAIN]PBX commands are implemented.
- the [GET|SET]SESSIONSETTINGS commands are implemented.

- the ROSTER and BILLING commands are documented.

## WebAdmin

- the Refresh Self-Signed button has been added to the TLS and S/MIME Domain Admin pages.
- Last Failed login information is displayed on the Account settings page. The Clear Failed Logins operation is implemented.
- Account rename operation now can move Accounts between Domains.
- the Telnum List page is implemented.
- the secondary logs browser is implemented.
- the Queue Message browser can show Message contents now.
- the DIALOG and MEDIA monitor pages are implemented.
- Domain Security Private Keys in the PKCS#8 format can be imported now.
- when switching Domain IP Addresses from some mode to Manual, the current IP addresses are pre-set.
- the Domain Object List includes Mailing List objects now.
- the BannerInfo and HiddenSkins Account settings are implemented.

## WebUser

- vCard PHOTO elements are displayed inline and can be modified now.
- Call Control settings pages now can upload and remove custom Music-on-Hold and RingBack files.
- the Finnish, Greek, Albanian languages have been added.
- the plain-text URL detection algorithm has been changed to allow for URLs containing the "()" and "[]" symbols.

## WebApp

- CG/PL Web Applications are supported now.

## WSSP

- the LENGTH prefix is implemented.
- the MAILBOXMENU structure now uses UTF8-encoded mailbox names as menu values.

## RADIUS

- trailing zero in the username attribute is ignored now.

## DIRECTORY

- "record limit" is now applied to cross-unit searches, too.

## FingerNotifier

- non-blocking connection opening with a short (3 seconds) time-out is used.

## Utils

- mailbox renaming options have been added to the MoveIMAPMail utility.

# CommuniGate Pro

| Introduction | Features | **History** | How To | Help Me |
|---|---|---|---|---|

# Version 5.1 Revision History

- **6.2**
- **6.1**
- **6.0**
- **5.4**
- **5.3**
- **5.2**
- **5.1: Summary**
- **5.0**
- **4.x**
- **3.x**
- **2.x**
- **1.x**

RSS

**5.1.16** 16-Mar-2008

`Valid Core License Keys`: issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- MAPI: the MAPI Connector version 1.51.16.1 is included.
- SMTP: 'fixed source IP address' separator has been changed from ':' to '|'.
- Bug Fix: Kernel: 5.1.15: "wildcard match" function could produce incorrect results.
- Bug Fix: RPOP: 3.0: incorrect data in RPOP.data file could crash the server on startup.

**5.1.15** 06-Mar-2008

`Valid Core License Keys`: issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- The Japanese Guide is included.
- MAPI: the MAPI Connector version 1.51.15.1 is included.
- Bug Fix: CLUSTER: 5.1c5: in Cluster, the load balancer addresses used the 5060 port for TLS, too.
- Bug Fix: LDAP: 5.1c1: search controls parsing failed if the "criticality" field was absent.
- Bug Fix: CLUSTER: 5.1c1: the SIP Farm settings were not stored in the settings file.
- Bug Fix: SMTP: 5.1c3: 'fixed source IP address' did not work for IPv6 addresses.
- Bug Fix: IMAP: 5.1.14: MAPI version check was broken, resulting in code re-downloading on every login.
- Bug Fix: IMAP: 4.1: certain SEARCH conditions could be "lost" causing broader search results and memory leaks.
- Bug Fix: IMAP: 4.2.0: the "separator discovery" LIST command did not return the "\Noselect" flag.
- Bug Fix: IMAP: 3.0: the "%"-sign in sample strings could be processed incorrectly.
- Bug Fix: IMAP: 3.0: the SEARCH operation did not ignore the time zone info in the Date: header field.
- Bug Fix: IMAP: 3.0: the SEARCH operation did not accept unknown KEYWORD

113

keys.
- Bug Fix: IMAP: 5.1.3: in some Mailbox multi-access scenarios "EXPUNGE" responses could be lost.
- Bug Fix: SMTP: 5.1c2: under heavy load, the server could crash while executing the shutdown sequence.
- Bug Fix: CLI: 5.1c2: moving an Account from a Directory based Domain to a different domain could crash the server.
- Bug Fix: Kernel: 5.1c2: a stream receiving a connection from some Debug IP address could put new connections into the "log all" mode, too.

**5.1.14** 22-Dec-2007

`Valid Core License Keys:` issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- MAPI: the MAPI Connector version 5.1.14 is included.
- Pronto: the Pronto version 5.1.14 is included.
- WebUser: the Korean language has been added.
- DNR: now "Dummy IP" addresses are detected in SRV/MX request responses, too.
- Bug Fix: LISTS: 5.1c5: mailing list owner Accounts could not be renamed.
- Bug Fix: WebAdmin: 5.1c4: the Cluster page had the Mailboxes and Cluster Admin setting controls swapped.
- Bug Fix: RULES: 5.1c1: the TimeOfDay=work condition could produce incorrect results.
- Bug Fix: MEDIAPROXY: 5.1c1: looped requests SDP "restoring" did not work for a client behind a far-end NAT.
- Bug Fix: CLI: 5.1c3: the GETTEMPCLIENTIPS command returned incorrect data.
- Bug Fix: ROSTER: 5.1c4: some RTC-specific operations could cause memory corruption.
- Bug Fix: CALLLEG: 5.1.3: canceling re-INVITE requests could result in "stuck" nodes.
- Bug Fix: CLUSTER: 5.1.1: when messages were rejected with "Submit Remotely" servers, no error message was returned.
- Bug Fix: CLUSTER: 4.0: "virtual" cluster Mailboxes incorrectly read message chunks larger than 1Mb.

**5.1.13** 23-Oct-2007

`Valid Core License Keys:` issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- Pronto: the Pronto Interface version 1.3 is included (SoftPhone beta included).
- Kernel: the New Zeeland time zone has been updated.
- Bug Fix: VCARD: 5.1.10: when a VCard 2.1 item was composed, non-structured field texts were escaped incorrectly.
- Bug Fix: Admin: 5.1c1: selecting all Enabled Services in Domain Setting effectively set the "default" value for those Services.
- Bug Fix: PBXLEG: 5.1c1: PRACK waiting timeout did not cause an immediate call failure.
- Bug Fix: PBXLEG: 5.1c2: when BreakBridge() operation is called and the breakBridge mode is "keep", media-less legs did not create Media channels.
- Bug Fix: LDAP: 5.1c3: when creating Account using LDAP Provisioning, the "uid" attribute was not removed from supplied LDAP attributes.
- Bug Fix: LDAP: 5.1c1: error messages in Server Log could be corrupted.
- Bug Fix: Foundation: 5.1c2: the --CGateApplication startup option did not work.
- Bug Fix: WebUser: when a Task was accepted, the generated Reply message did not copy the PERCENT-COMPLETE field value.
- Bug Fix: WebUser: when a Draft message was opened for editing, names of already attached files were shown undecoded.
- Bug Fix: SNMP: 3.0: MIB ACCESS clauses changed to MAX-ACCESS
- Bug Fix: DOMAIN: 5.1c4: Directory-based Domains did not load their Domain Rules upon restart.

**5.1.12** 29-Aug-2007

issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- Pronto: the Pronto Interface version 1.2 is included.
- Kernel: the new Venezuela time zone (-0430) has been added.
- Bug Fix: CLI: 5.1.6: the *DomainSkin commands could crash the server while checking user credentials.
- Bug Fix: SIGNAL: 5.1c1: "no answer" processing was different from "no registration" processing.
- Bug Fix: SIGNAL: 5.1c4: if a 200 response was received when other forked branches were pending, the Rule engine could be started again.
- Bug Fix: IMAP/ACAP: 4.2: malformed Q-strings could crash the server.

**5.1.11** 22-Jul-2007
`Valid Core License Keys:` issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- MAPI: the MAPI Connector version 1.2.12 is included.
- WebUser: now the HTML "AREA" tags are processed as "A" tags
- Foundation: IP read buffer under Solaris has been increased to allow for 1000+ local IPv4 addresses.
- Bug Fix: XIMSS: 5.1c2: updating Calendar Events with attachments could fail when sending update requests.
- Bug Fix: XIMSS: 5.1c3: the prefsStore operation did not update the internally-used preference set (re-login was required).
- Bug Fix: XIMSS: 5.1c2: certain race conditions during session closing could crash the server.
- Bug Fix: PBXLEG: 5.1.5: incoming INVITEs w/o SDP could result in abandoned leg nodes.
- Bug Fix: PBXLEG: 5.1.10: canceling and redirecting an incoming call could result in a BYE request being sent to the caller.
- Bug Fix: SIP: 5.1.6: the P-CGP-Private data element was not added to outgoing packets
- Bug Fix: SIGNAL: 5.1c4: the "MakeCall" function could use incorrectly URL-encoding
- Bug Fix: SIGNAL: 5.1.7: 408/480 error conditions did not start the 'delayed' Rule actions.
- Bug Fix: SIP: 5.1.2: response VIA parser incorrectly processed 'instance' (Rxxx) elements.
- Bug Fix: DOMAINS: 5.1c2: Directory-based Domain rules were stored incorrectly.
- Bug Fix: MEDIA: 5.1c3: rejecting a bad SDP in a re-INVITE and detecting a new/incorrect stream codec could cause a server crash.

**5.1.10** 19-Jun-2007
`Valid Core License Keys:` issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- Multilingual Guide system is implemented.
- The Russian Guide is included.
- Pronto: the Pronto Interface version 1.1 is included.
- MAPI: the MAPI Connector version 1.2.11 is included.
- XIMSS: index-based and range-type message sets are implemented, the folderBrowse parameters have changed.
- XIMSS: the fileCopy operation has been implemented.
- IMAP: the MOVE commands are implemented.
- IMAP: the ENCRYPTED parameter is documented.
- Kernel: now bracket-comments are rejected in E-mail addresses that are not supposed to have comments (SMTP addresses, certificate E-mails, etc.)
- Router: the Domain-level Access To Unknown settings are implemented.
- Router: Table records can have more than one Mail/Signal/Access prefix.
- FTP: file/directory name limit is increased to 1024 bytes.
- ACCOUNT: now Initial Message text in Account Template can contain macro-symbols.
- AUTH: the Digest "uri" parameter size limit has been increased.
- WebUser: the Ukrainian language has been added.
- WebUser: the number of Mailbox Viewer fields a user can specify is limited now.
- WebUser: now Tasks are sorted by their Due dates, and then - by their modification dates.
- CALENDAR: now "exception" events publishing does not conflict with the main event publishing.
- Bug Fix: WebUser: 4.1: now the HTML clean-up routine correctly processes ISO-2022 data in tag parameter

values, too.

- Bug Fix: vCard: 4.1: when a vCard text was generated, some special symbols were not properly escaped.
- Bug Fix: SIGNAL: 5.1c2: watcher-info NOTIFY data contained subscriber's Contact data instead of AOR (From:) data.
- Bug Fix: Foundation: 5.1c1: binary zeros in XML data elements and attributes could result in mis-formed XML.
- Bug Fix: SNMP: 5.1c3: the autogenerated MIB file had a syntax error.
- Bug Fix: Admin: 5.1c1: when moving an Account to a different Domain, Group and Telnum information was updated incorrectly.
- Bug Fix: Skins: 5.1.8: the first (after restart) file uploading operation could fail on Server-wide and Cluster-wide Skins.
- Bug Fix: CALLLEG: 5.1.8: BYE signals were not sent on certain communication errors causing logical disconnect.
- Bug Fix: CALLLEG: 5.1.8: explicitly delayed BYE signals could crash the server.

**5.1.9** 11-May-2007

`Valid Core License Keys:` issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- Pronto: the Pronto Interface version 1.0 is included.
- MAPI: the MAPI Connector version 1.2.10 is included.
- Kernel: the "server zone" is renamed from "***" to "HostOS" and turned into a "real" time-zone.
- CLI: the UPDATEWEBUSERDEFAULTS command is implemented.
- S/MIME: signature certificate authority chains are processed now.
- SMTP: when non-220 initial prompt is received, the QUIT command is sent before disconnecting.
- Files: .meta-files are implemented.
- WebUser: now html-cleanup processes all standard attributes with URI-type values.
- WebUser: now html-cleanup process removes scripting tags together with their contents.
- WebUser: now USENET signatures ("-- " lines) are processed when rendering "format=flowed" messages.
- SIP: the Send P-Asserted-Identity option is implemented.
- LDAP: the "anr" pseudo-attribute is supported.
- TFTP: the Run Sessions on Controller option is implemented.
- FTP: the Passive Mode setting is implemented.
- XIMSS: vCard xml format has changed.
- XMPP: the XEP-0077 (Registration) is implemented.
- TLS: wildcard certificates are accepted now.
- Bug Fix: WebSkins: 5.1c2: the "parentSkin" string element did not work for Domain-level Skins.
- Bug Fix: DIRECTORY: 5.1.8: objectClass fast-search could result in mis-formed DNs.
- Bug Fix: FTP: 5.1c3: active-mode FTP may fail in some Cluster configurations.
- Bug Fix: CLUSTER: 5.1.2: messages from Virtual Mailboxes could get redirected using incorrect EOLs.
- Bug Fix: WEBUSER: 5.1c3: under some conditions, cookie checks did not work.
- Bug Fix: SIGNAL: 5.1c1: "sips" requests could be converted to "sip" requests on forking.
- Bug Fix: SIP: 5.1c2: packets with empty Subject fields were rejected.
- Bug Fix: Skins: 5.1c6: directory for Server-wide Skins was not created automatically.
- Bug Fix: ROSTER: 5.1.c3: group creation did not work after 9 groups being created.
- Bug Fix: MIME: 4.0: MIME-header composer could incorrectly process some shift-jis strings.
- Bug Fix: RULES: 5.0.1: malformed Rules could crash the server.
- Bug Fix: RULES: 5.1c3: the Write To Log operation did not process macro symbols.
- Bug Fix: HTTP: 4.3: routing requests to an Account Storage subdirectory could cause memory corruption.
- Bug Fix: HTTP: 5.1.8: using malformed Personal Web Site URIs could case memory corruption.
- Bug Fix: WebAdmin: 5.1c1: some Domain Settings did not contain "default" values if the Server-wide Domain Defaults were never set.
- Bug Fix: EVENTS: 5.1c1: watcher notifications incorrectly showed subscription states.
- Bug Fix: XIMSS: 5.1c2: malformed E-mail header fields could result in malformed XML data messages.

**5.1.8** 30-Mar-2007

`Valid Core License Keys`: issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- Pronto: the Pronto Interface version 0.8.9 is included.
- MAPI: the MAPI Connector version 1.2.9 is included.
- Kernel: a workaround for MacOS X library bug corrupting memory under heavy load is implemented.
- SIGNAL: the Flow Control options are implemented.
- SECURITY: the Domain S/MIME Escrow Certificate support is implemented.
- CLUSTER: the heartbeat timeout decreased to 30 seconds.
- DIRECTORY: binary attributes (such as certificates) in Remote Units are sent with the ";binary" suffix (if the LDAP protocol version is set to 3).
- DIRECTORY: unauthenticated requests match only Access Rights specified for "anyone" Bind DN.
- WSSP: the FOREACHREV element is implemented.
- SIGNALRULES: if an account has zero registrations, its first "staged" Rule set is applied to incoming calls.
- PBX: bridged provisioning responses are "stacked" to compensate for different egress and ingress signaling speeds.
- CALLLEG: when call is disconnected, a small delay is introduced between CANCEL or reINVITE-response and the closing BYE request.
- WebAdmin: the disabled Mobile service is now ignored for the Main Domain only.
- SIGNALRULES: the AgentType (User-Agent) condition is implemented.
- TFTP: if a file name contains a leading slash and no other slash symbols, the leading slash is removed.
- SIP: packets with multipart bodies with an SDP sub-part are supported now.
- SIP: the UDP TOS Tag option is implemented.
- WebUser: now the From columns are replaced with To columns not only for Send and Draft Mailboxes, but for their sub-mailboxes, too.
- XIMSS: the makeCall operation is implemented, the messageAppend and appendVCard operations support folder parameter now.
- MAILBOX: UTF-7 validity of new Mailbox names is checked now.
- Bug Fix: XIMSS: 5.1.7: the "session" message attribute "userName" was sent as "username"
- Bug Fix: XIMSS: 5.1c1: the *Box preferences were not converted to and from the UTF-7 Mailbox name encoding.
- Bug Fix: XIMSS: 5.1c4: when adding the E-mail parts using the "UploadID" attribute, 2 different Content-Transfer-Encoding fields were created.
- Bug Fix: CLUSTER: 4.2: attempts to access Personal File Storage using other Account credentials could fail in Cluster.
- Bug Fix: MAILBOX: 4.0: now Mailbox name components cannot start with "." to avoid creating "invisible" Mailboxes.
- Bug Fix: Kernel: 5.1c2: Digest challenges with multiple-choice "qop" parameters were processed incorrectly.
- Bug Fix: DNS: 5.1c2: excessive number of MX/SRV records could result in read access attempts outside the server memory.
- Bug Fix: CLUSTER: 5.1.6: controller could crash on startup if the deprecated --ClusterController option was used.
- Bug Fix: SIP: 5.1c2: old-style INVITE and ACK-on-negative requests could be not matched in a transaction search.
- Bug Fix: Admin: 5.1c1: the ListObjects functions did not retrieve Forwarders in Directory-based Domains.
- Bug Fix: Rules: 5.0c2: Rules with names starting with '*' could not be modified or removed.
- Bug Fix: HTTP: 5.0c3: incorrect CalendarData access could crash the server.

**5.1.7** 25-Feb-2007
`Valid Core License Keys`: issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- Client: the [Media Plugin](#) 1.0.1 is released.
- MAPI: the MAPI Connector version 1.2.7 is included: GAL support, Outlook 2007 support.
- Pronto: the Pronto Interface version 0.8.7 is included.
- WebUser: if the ReplyHeader Preference is an empty string, the original message text is not added to reply

messages.

- WebUser: when a message with an empty body is being stored as a Draft, the message body part is not omitted, so the Edit Draft operation does not try to edit the first attachment.
- UserSessions: the session timeout management has changed to avoid problems with XIMSS sessions.
- TLS: now the certificate subjectAltName elements can be used for Certificate-base authentication.
- HTTPO: if the returned data is a text in a non-UTF8 charset, the returned data is converted into UTF-8.
- SIP: incorrectly formed Refer-To fields (Mediatrix) are accepted now.
- SMTP: an option to start Parallel Channels for the specified queue size is implemented.
- XIMSS: the mailboxClass parameter is added to the mailboxList command.
- XIMSS: the recoverPassword, spellerList, spellerCheck commands are implemented.
- SIGNAL: supplementary 200-INVITE responses are upstreamed now.
- CallLeg: PRACK processing changed to avoid cross-wiring with 200-INVITE in B2BUA applications.
- CallLeg: dialogs started with supplementary 2xx-INVITE responses are gracefully terminated now.
- CallLeg: 2xx-INVITE responses without SDP are supported now.
- PBX: now re-INVITEs sent after BridgeBreak contain an offer with all server-supported codecs.
- WebUser: calendaring page processors are completely converted to process data in local-time terms.
- SDP: now symbolic names are accepted in the "o" lines.
- RADIUS: now more error conditions result in negative responses instead of request dropping.
- Kernel: the DIGEST/NTLM parameter parser modified to accept more white-space separators.
- Kernel: now case-insensitive "wildcard" search operations support non-Latin symbols, too.
- Bug Fix: DNR: 5.1c1: NAPTR processing used "POSIX Basic" Regular Expressions instead of "POSIX Extended" ones.
- Bug Fix: WebAdmin: 5.0.3: opening the Alerts page for an non-existent account could crash the server.
- Bug Fix: Mailbox: 5.1.6: file names for mdir-type Mailboxes messages with new message flags were calculated incorrectly.
- Bug Fix: CLUSTER: 5.1c3: UDP response packets relayed via Cluster members were not directed to the Via: addresses.
- Bug Fix: WebUser: 5.1.6: search filters in Directory Address books worked incorrectly.
- Bug Fix: Queue: 5.1.2: generating a log record for a certain error condition could crash the server.
- Bug Fix: WebSkins: 5.1c.3: the HTML editor in Simplex-based Skins could be made execute arbitrary script code.
- Bug Fix: CallLeg: 5.1c2: when a session was interrupted because of refresh issues, a BYE request was not sent.
- Bug Fix: CallLeg: 5.1c3: the NOTIFY requests generated for REFER contacted the Subscription-State header in the final requests only.
- Bug Fix: CG/PL: 5.1c1: incorrect "openMailbox" call parameters could crash the server.
- Bug Fix: XIMSS: 5.1.5: <folderMessage> "e-field" sub-elements were mis-formed.
- Bug Fix: WebAdmin: 5.1c6: the "other" options did not exist on the Trigger Events page.
- Bug Fix: LIST: 5.1.2: the "listserver"-generated reports were discarded without delivery.
- Bug Fix: WebUser: 5.1c2: the Enable Vacation button did not clear the "replied" list.

## 5.1.6 02-Feb-2007
Valid Core License Keys: issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- Pronto: the Pronto Interface version 0.8.6 is included.
- MAILBOX: the Junk, $Label1, $Label2, $Label3 message flags are available now. In TextMailboxes these new flags can be set only for the messages stored with the latest 5.1 versions.
- DIRECTORY: the stringMatch search operation is redesigned.
- LOG: now Crash-level records can be processed with Trigger Handlers.
- CALLLEG: the StartCall function now supports the P-CGP-Private parameter.
- CALLLEG: responses to INVITE with Supported:timer now contain the Require:timer field.
- Admin: the Signal "Service" setting is implemented.
- SIP: URI and field parameter value escaping methods have been improved.
- SIP: a workaround for several gateways and UA: if a 18x response was sent, a locally-generated final response now reuses the same To-tag.

- PBX: the SetInterruptOnDTMF function is implemented.
- PBX: the SetBridgedProvision function is implemented.
- XMPP: the non-standard Jabber "digest" authentication method can be disabled by switching off the CRAM-MD5 method.
- TLS: the SSL 3.0 "no_certificates" Alert message is supported now (Firefox 2.0 may issue it when a client-side certificate is requested).
- TLS: the Abort on Wrong Client Certificate option is implemented.
- XIMSS, XMPP: Subject processing added, "typing" processing changed.
- XIMSS: Mailbox sorting algorithms have been improved.
- TFTP: the Try IP-Address Directory option is implemented.
- SignalRules: the Active Devices condition is implemented.
- WebUser: the Polish language has been added.
- Utils: the MoveIMAPMail command now copies all message flags supported with the target server.
- Bug Fix: CLI: 5.1.4: the UPDATEACCOUNT[MAIL|SIGNAL]RULE command syntax has been changed to avoid "string merging" (especially in the inter-cluster protocol)
- Bug Fix: CLI: 5.1.4: the UPDATEACCOUNT[MAIL|SIGNAL]RULE commands did not add new rules.
- Bug Fix: DIRECTORY: 5.1c2: 'fast search' did not work for the objectClass attribute.
- Bug Fix: PBX: 5.1: redirectCall operation did not reset the Task status.
- Bug Fix: SIP: 5.1.5: epid processing was incompatible with some versions of Polycom phones.
- Bug Fix: SMTP: 5.1c1: authenticating on "native" backends did not work if the --ForeignStatic option was enabled.
- Bug Fix: SMTP: 5.1c1: the DIGEST-MD5 authentication did not work in Cluster
- Bug Fix: POP: 5.1c1: the DIGEST-MD5 authentication did not work in Cluster
- Bug Fix: CALLEG: 5.1c1: the Redirect operation and incoming Cancel did not reset the "halfConnected" state.
- Bug Fix: SDP: 5.1c2: the copy routine did not copy the "ptime" attribute.
- Bug Fix: RADIUS: 5.1c3: the RADIUSCall operation (CG/PL) did not pad the password data with zeros.
- Bug Fix: LDAP: 5.0.2: the one-level search operations incorrectly composed Unit root DNs.

## 5.1.5 15-Jan-2007
`Valid Core License Keys`: issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- Pronto: the Pronto Interface version 0.8.5 is included.
- MAILBOX: the number of messages with the Service flag is stored in Mailbox meta-data now.
- WebUser: now Service messages are not included into the message counter displayed on the "mailboxes" page.
- DIRECTORY: the 'displayName' processing moved to the Directory 'Local Units'.
- XIMSS: E-fields (E-To, E-From, etc) can be specified in the folderOpen operations.
- XIMSS: the mailboxSubList, mailboxSubUpdate operations are implemented.
- XIMSS: the mailboxRightsGet, mailboxACLList, and mailboxACLUpdate operations are implemented.
- Kernel: the {SSHA} password encryption method is supported.
- Bug Fix: Directory: 5.1.4: the last record attribute could not be retrieved in some search operations.
- Bug Fix: SMTP: 5.1c2: Group and LIST addresses in MAIL FROM: field did not require authentication when Force AUTH option was enabled.
- Bug Fix: FILTER: 3.0: removing an active Queue Filter can crash the server under heavy load.
- Bug Fix: WebAdmin: 5.1c2: some standard PSTN server-wide Account Default settings could not be set to empty strings.
- Bug Fix: SIGNALRULES: 5.1c2: the "Blocked" Rule could be composed incorrectly.
- Bug Fix: WebUser: 5.1c2: the Remove Attachments button was displayed even when attachments were un-removable.
- Bug Fix: WebUser: 5.1c1: the "Prev Week" link in the Calendar Weekly view jumped back by 2 weeks instead of 1.
- Bug Fix: ACAP: 5.1.4: the AUTHENTICATION challenge responses were processed incorrectly.
- Bug Fix: Platform: 5.1c1: not all local IP aliases were detected under AIX.
- Bug Fix: TFTP: 4.3: certain retrieval operations could cause memory corruption.
- Bug Fix: CLI: 5.1.4: the UPDATEACCOUNT[MAIL|SIGNAL]RULE commands did not accept string-type

parameters.
- Bug Fix: WebUser/XIMSS: 5.1c2: in some calendar views, all-day events could be displayed spanning for one extra day.

**5.1.4** 22-Dec-06
`Valid Core License Keys:` issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- Kernel: secondary TCP channels now can use Non-Blocking sockets, too.
- Kernel: multi-line RFC2231 fields are supported now.
- Kernel: E-mail usernames can now have a leading "!" symbol.
- Kernel: NorthAmerica/* time zone descriptors are updated to reflect the changes in USA DST.
- Kernel: Kerberos NT-SMTP-NAME and NT-ENTERPRISE are supported now.
- MAPI: the MAPI Connector version 1.2.5 is included.
- Pronto: the Pronto Interface version 0.8.4 is included.
- Security: delays after login failures have been moved "up" to the module components to decrease resource consumption.
- WebAdmin: the Interface Type (Expertise Level) concept is implemented.
- WebAdmin: the Access and Service module settings are specified on separate pages now.
- Admin: the PSTN Settings Domain Access Right is implemented.
- HTTP: the HTTP settings have been moved from the WebUser page to the HTTP module pages.
- MAILBOX: locking operations can time-out now. The time-out is set to 30 seconds.
- ACAP: literals in AUTHENTICATION challenge responses are accepted now.
- SIGNAL: the "must proxy" flags are now set directly in SDP, so they are preserved when a call travels between protocols and tasks.
- CALLLEG: the StartCall parameters are remembered and reused when REFER'ed INVITE requests are composed.
- CLI: the GETSERVERTRUSTEDCERTS, SETSERVERTRUSTEDCERTS, GETCLUSTERTRUSTEDCERTS, SETCLUSTERTRUSTEDCERTS commands are documented.
- CLI: the [GET|SET][SERVER|CLUSTER|DOMAIN|ACCOUNT]RULES commands are renamed into [GET|SET][SERVER|CLUSTER|DOMAIN|ACCOUNT]MAILRULES. Old commands are still supported.
- CLI: the [GET|SET][SERVER|CLUSTER|DOMAIN|ACCOUNT]SIGNALRULES commands are documented.
- CLI: the UPDATEACCOUNT[MAIL|SIGNAL]RULE commands are documented.
- PBX: the Transfer and CallInfo functions are documented.
- CG/PL: the UPDATEACCOUNT[MAIL|SIGNAL]RULE functions are documented.
- SIGNAL: the Router supports the special "incomplete" address now. Signals routed to that address are rejected with the 484 Address Incomplete error.
- XIMSS: the listDirectory operation is implemented.
- Bug Fix: LDAP: 5.1c.1: the filter string compiler incorrectly processed leading '*' symbols.
- Bug Fix: PBX: 5.1.3: default audio codecs were not set for SDP with both audio and video channels.
- Bug Fix: VoiceMail: 5.1.2: if the HomePSTNNumber setting was set to an empty string, all incoming calls were processed as "home calls".
- Bug Fix: MAILBOX: 5.0c.3: BSD Mailbox separator lines in the 'new' format (used for media messages) contained a trailing binary zero.
- Bug Fix: SIP Server: 5.1c1: duplicated malformed packets composed with MS RTC could cause server crashes under heavy load.
- Bug Fix: DIRECTORY: 5.1c3: search operations processed absent attributes as empty strings.
- Bug Fix: NATPING: 5.1.3: incorrect log records were produced.
- Bug Fix: USERSESSION: 5.1.3: the makeCall function did not work.
- Bug Fix: ROUTER: 5.1.3: the 'internal loop' situation could put garbage into the Log.
- Bug Fix: Telnums: 5.1c1: on restart, telnums in one domain could conflict with deleted telnums in some other domain.
- Bug Fix: SIP: 5.1c5: postponed CANCEL requests were not sent if the transaction was disconnected.
- Bug Fix: SIP: 5.1c1: request modification when generating negative responses could cause memory corruption.
- Bug Fix: WebAdmin: 5.1c5: the Directory pages did not allow to open non-root Unit Schema.

- Bug Fix: CLUSTER: 5.1c3: GSSAPI/KERBEROS, EXTERNAL, and IMPERSONATE login methods did not work for session-based protocols.
- Bug Fix: Foundation: 5.1c2: the TOS byte could not be set on some platforms.
- Bug Fix: DIRECTORY: 5.1c1: search optimization could result in reporting incorrect DNs for records found.

**5.1.3** 25-Nov-06

`Valid Core License Keys:` issued between 01-Oct-2004 and 31-Oct-2004*, or on or after 01-Oct-2005.

- SMTP: if a receiving party breaks a connection after receiving some messages, a new connection is opened immediately.
- XIMSS: synchronous data messages now include the same "id" attribute as the request messages.
- XIMSS: the cliExecute command is implemented.
- XIMSS: stream encryption is implemented.
- MEDIA: partial External Transcoder support is implemented.
- MAPI: the version 1.2.5 of the MAPI Connector is included.
- TLS: connections from TLS 1.1 (and higher) clients are accepted now.
- ENQUEUER: log format has been changed.
- LDAP: Attribute ";binary" option is processed now.
- DIRECTORY: Remote Units now understand the ";binary" option in returned attributes.
- STATISTICS: MIB ObjectIDs for SMTP and SIP elements have been changed.
- CG/PL: the READTELNUMS function is implemented.
- EVENT: the SUSBCRIBE requests with the same CallID and From-tag clean the existing matching subscriptions.
- Presence: RFC4480 (extended PIDF) is supported now.
- CallLeg: the Allow/Supported fields are included into 200-INVITE responses.
- Bug Fix: XIMSS: 5.1c5: signalBind media address was not remembered. An unnecessary media proxy could be created.
- Bug Fix: CLI: 4.3c3: the FindWebUserSession command could delay the WebUser Session timeouter.
- Bug Fix: Presence: 5.1c4: "dead" MS RTC presence clients could cause problems for other MS RTC clients registered with the same Account.
- Bug Fix: XMPP: 5.1c2: Roster element 'subscription=' values could be reported incorrectly.
- Bug Fix: WSSP: 5.0c2: when HTMLSUBSTMAILBOX function was used with non-UTF8 pages, Mailbox name charset could be converted twice.
- Bug Fix: SMTP: 5.1c3: ATRN retrieval did not work.
- Bug Fix: ROUTER: 5.0c3: All-Local Routing Records could leave the secondary domain name intact, causing Router loops.
- Bug Fix: PBX: 5.1c4: race conditions on task launch could crash the server.
- Bug Fix: RPOP: 5.1c3: modifying host/account names in WebAdmin could reset the RPOP password data.
- Bug Fix: XIMSS: 5.1c4: the signalCode attribute in asynchronous call* messages was malformed.
- Bug Fix: WebAdmin: 5.1c2: the SMTP 'Hold' and 'Send Encrypted' domain lists could not be cleared.
- Bug Fix: WebUser: 5.1c6: displaying attachments with certain names could crash the server.
- Bug Fix: Trigger: 5.1c4: the SendIM action sent IMs with an incorrect "From" address.
- Bug Fix: Foundation: 5.1.2: DNS Resolver did not work on some IPv6 systems.
- Bug Fix: SIP: 5.1c4: If a From/To/Contact field did not contain '<', '>', parameters were processed as URI parameters rather than field parameters.
- Bug Fix: PBX: 5.1c2: provisionCall(true,true) did not wake up after receiving PRACK.
- Bug Fix: MAPI: 5.1c1: the SETACCOUNTRULES operation could filter our the mail rules the user was allowed to created.
- Bug Fix: Listener: 5.1c2: the Initial TLS setting could not be reset without re-creating the socket.
- Bug Fix: CLUSTER: 5.1c3: WebAdmin requests were passed to backends with modified Host field causing HTTP-redirect failures.
- Bug Fix: CLUSTER: 5.1c4: certain WebAdmin pages did not use Cluster-wide settings.
- Bug Fix: CLUSTER: 4.0: recovery after detecting an over-the-limit inter-cluster message caused protocol synching problems.

- Bug Fix: CLUSTER: 5.1c1: TELNUMs did not work in a Dynamic Cluster.

**5.1.2** 05-Nov-06
`Valid Core License Keys:` issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- XIMSS: the Automated Rule management operations are documented.
- WSSP: the TIMEMENU units can specify menu elements using time suffixes.
- WSSP: the SIZEMENU units can specify menu elements using size suffixes.
- SIP: the Request URI scanning algorithm is improved.
- PBX: now Real-Time application parameters are always presented as the Vars().startParameter array.
- XMPP: now the secure 5223 port is enabled by default.
- Signal: request field parameters are processed as case-sensitive now.
- Admin: new installations now use A-crpt password encryption by default.
- WebUser: the WebUserMethod option can be used to automate WebUser logins.
- Bug Fix: HTTP: 5.1c1: the '+' symbols in the x-www-form-urlencoded request bodies were not converted into spaces.
- Bug Fix: Presence: 5.1c2: if the Presence Subscribe contained only the Accept: application/cpim-pidf+xml, the notifications were sent using the application/xpidf+xml format.
- Bug Fix: XIMSS: 5.1c1: 'auth' challenge responses were processed incorrectly.
- Bug Fix: SIP: 5.1c1: "Presence Subscribe on Register" workaround used the same string for To- and From-tags in the SUBSCRIBE requests.
- Bug Fix: Presence: 5.1c5: pidf+xml formats were processed incorrectly.
- Bug Fix: XMPP, XIMSS: 5.1c3: contact Group modifications were not reported to the client.
- Bug Fix: Domain: 4.3: the number of domain forwarders could be calculated incorrectly after restart.
- Bug Fix: WebAdmin: 5.1c4: PSTN "password"-type Settings could be reset if Server/Cluster-wide Default Account Settings were modified.
- Bug Fix: Foundation: 5.1c3: XML character references were decoded incorrectly.

**5.1.1** 22-Oct-06
`Valid Core License Keys:` issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- XIMSS: the Calendaring, File Storage, and Task management operations are documented.
- XIMSS: many operations are renamed - the old names still can be used, but XIMSS applications should be updated before 5.2 release.
- SIGNAL: only the presence SUBSCRIBE with the adrl+xml content subtype are processed as "self-subscriptions".
- Admin: the Log Display Size Limit Preference option is implemented.
- SIP: strict-route processing moved from SIP Server to SIP Client.
- Bug Fix: WebUser: 5.0c1: if the login page was opened using the wml/imode/imodejp references, the response Content-Type was not set.
- Bug Fix: WebAdmin: 5.1.0: updating any Settings->General page put incorrect data into the Main.settings file, making it unreadable by 5.1.0 after server restart.
- Bug Fix: WebAdmin: 5.1.0: the MIB link did not work.
- Bug Fix: Admin: 5.1.0: if a postmaster account is replaced with a different object (Forwarder, Alias, etc.), the server shut down in 10 minutes due to "no postmaster password" situation.
- Bug Fix: Platform: 5.1.0: a bug in Microsoft Visual C++ optimizing compiler caused various problems in the Win32/Intel version.
- Bug Fix: MEDIAPROXY: 5.1c4: re-INVITE operation did not reset the "remote port" data, causing communication problems for clients that switch ports during one call (such as voice/fax gateways).

**5.1.0** 15-Oct-06
`Valid Core License Keys:` issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- Admin: Initial Installation procedure has changed: the initial postmaster password is set via the WebAdmin

Interface.

- WebAdmin: the Separate WebAdmin Realms option is implemented. By default the WebAdmin Interface now uses one authentication realm.
- IMAP: if an 8-bit symbol is met in an e-mail address part, the FETCH ENVELOPE element now presents that part as a literal.
- MEDIA: the Source Port Restriction option is implemented.
- MEDIA: codec processing has changed (to support stateful codecs).
- WSSP: the FOREACHINC element is documented.
- WebAdmin: "in-line" (no iframe) Log Viewer Preference setting is implemented.
- CALLLEG: deadlocks are detected when both sides of a bridged call try to re-INVITE at the same time.
- Bug Fix: CLUSTER: 5.1c4: unnecessary inter-cluster media relays could be built.
- Bug Fix: CLUSTER: 5.0c2: PBX environment management had file transfer problems.
- Bug Fix: DNR: 5.1c1: MX and SRV records with the same priority were processed incorrectly.
- Bug Fix: DNR: 5.1c6: PUBLISH for an unknown event package could crash the server.
- Bug Fix: XIMSS/XMPP: 5.1c5: IM receiving did not work.

**5.1c6** 10-Oct-06

`Valid Core License Keys:` issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- Rules: the Submit Address condition is now supported in Signal Rules, too.
- SIP: the TimerB setting is implemented.
- SIP: now negative-ACK requests contain the same authentication fields as the failed INVITEs.
- SIP: PUBLISH requests w/o Expires: header field are accepted now.
- PBX: the TransferCall function is documented.
- PBX: SendDTMF() operation sends INFO-requests if the call is actually bridged.
- PBXApp: the Alternative Number and the Home PSTN Number settings are implemented.
- PBXApp: the "mailbox service" application now supports voicemail forwarding.
- MIME: RFC2231 language/charset tags are processed now.
- MEDIAPROXY: the sending addresses are built using "other side Media IP" whenever it is known.
- CG/PL: MailboxRedirectByUID, MailboxForwardByUID functions are implemented.
- XIMSS: redirectMessage, forwardMessage, confirmMessage operations are implemented.
- Foundation: the TIS-620 alias for ISO-8859-11 is implemented.
- WebAdmin: requests to the server-specific pages in a Dynamic Cluster are redirected to the Cluster realm.
- CG/PL: now the ReadGroupMembers function can access Groups in other Domains.
- CG/PL: the RADIUSCall function now supports vendor-specific attributes.
- RADIUS: vendor-specific attributes are supported now.
- RADIUS: External Helper now gets all request attributes and an additional Password attribute containing the "shared secret" string.
- CLUSTER: support for "stateful" UDP load balancers has been added.
- WSSP: the ^ (XOR) operation and the BOOLARRAY() function are implemented.
- Bug Fix: WebAdmin: 5.1c5: authentication in secondary domains and referenced cluster members was processed incorrectly.
- Bug Fix: WebAdmin: 5.1c5: LIST module links were broken.
- Bug Fix: WebAdmin: 5.1c5: Domain sub-skins and PBX language links were broken.
- Bug Fix: WebAdmin: 5.1c2: custom PSTN settings could not be set.
- Bug Fix: SIP: 5.1c1: MS RTC signing did not work if the From: field contained URI-parameters.
- Bug Fix: CG/PL: 5.1c1: the routeAddress() function with a non-string parameter crashed the server.
- Bug Fix: UDP Listener: 5.1c1: some UDP Listener settings from the previous versions could be lost.
- Bug Fix: Kernel: 5.1c1: some 2-byte charset names were not recognized.
- Bug Fix: CLI: 5.1c1: GETCURRENTCONTROLLER operation returned the loopback address when it was executed on the controller itself.
- Bug Fix: PBX: 5.1c1: the localareacall.sppr application had compile-time errors.
- Bug Fix: SIP: 4.2: corrupted TCP messages could crash the server.

**5.1c5** 01-Oct-06
`Valid Core License Keys:` issued between 01-Oct-2004 and 31-Oct-2004, or on or after 01-Oct-2005.

- WebAdmin: the WebAdmin Interface has been redesigned.
- Log: the Open Showing Last option has been moved to Admin Preferences.
- HTTPOutput: "chunked" responses are supported now.
- XMPP: Jabber "digest" authentication is supported now.
- XMPP: the 'stream' 'to' parameters are now used to set the target domain (for TLS negotiations and logging in).
- MediaServer: the A/V socket TOS byte can be set now.
- Signal: the "Auth all outgoing INVITEs" option is implemented.
- CLUSTER: Signal: the authenticated/redirector information is passed between Cluster members.
- CLUSTER: "nonce" distribution algorithm has been improved.
- CLUSTER: XMPP and XIMSS support has been implemented.
- MAILBOX: the BSD Mailbox "restore damaged counter" procedure is extended to index files.
- MEDIA: All-Info logging is supported for media streams to/from Debug IP Addresses.
- SIP: the "Send 100-Trying" options are implemented.
- SIP: CANCEL requests are postponed till the first provisioning response.
- CALLLEG: refresh requests are now sent 1/2 into the refresh interval.
- CALLLEG: refresh requests are considered failed on 408 and 481 codes only.
- WebUser: the Language parameters are "cleaned" to disable URL-embedded scripts.
- WebUser: the Call Log viewer is implemented.
- Bug Fix: Foundation: 5.1c4: the XML parser incorrectly processed CDATA blocks.
- Bug Fix: CLI: 5.1c2: the KILLACCOUNTSESSIONS command was not accepted.
- Bug Fix: CLUSTER: 5.1c2: XMPP login did not work in Cluster.
- Bug Fix: CLUSTER: 5.1c2: Domain Signal Rules updates were not propagated through the Cluster.
- Bug Fix: CG/PL: 5.0c5: readGroup() with a non-string parameter could crash the server.
- Bug Fix: CLUSTER: 4.1: OS/2 or Windows-based frontends could corrupt messages during inter-cluster mail delivery.

**5.1c4** 12-Sep-06
`Valid Core License Keys:` issued between 01-Sep-2004 and 31-Oct-2004, or on or after 01-Sep-2005.

- Pronto: a beta version of the Pronto! Flash-based interface is included.
- Platform: Linux/s390x (64-bit) version is released.
- Platform: MacOSX: the startup scripts are moved from the /System/Library to the /Library directory.
- WebAdmin: the WebAdmin Interface has been switched to Skins. The WebAdmin application directory has been removed.
- SIGNAL: the RTC (Windows Messenger) <-> Roster integration is implemented.
- SIGNAL: Request URI parameters are preserved when a request is routed to a remote domain.
- MEDIAPROXY: "collapse proxy" processing has been modified.
- DIRECTORY: search operations now support multi-value attributes.
- CALLLEG: the uri-parameters (such as user=phone) are now recorded into the dialog context.
- CALLLEG: on a 422 response, the INVITE request is resent with the returned Min-SE value.
- NETWORK: UDP Listeners support multiple sockets now.
- ASYNCOBJECTS: dying async objects now process pending event messages.
- PBX: the StartBridgedCall function is implemented (it is used to implement "transparent bridging" for B2BUA applications).
- CG/PL: the StoreCDR procedure is implemented.
- XIMSS: behind-NAT audio clients are supported.
- XIMSS: the readCalendar operation is implemented.
- XIMSS: the copyMIME element is implemented.
- XIMSS: the Roster and Presence operations are implemented.
- XIMSS: the retrieveXML operation is implemented.
- XIMSS: the callTransfer, callSendDTMF operations are implemented.

- MAPI: the version 1.2.3 of the MAPI Connector is included.
- CLUSTER: the information about the packet target address is passed within the SIP Farm.
- WebUser: the Previous Login and the Last Failed Login information is available on the "hello" (Summary) page.
- LDAP: special processing for the displayName attribute is extended to the GE and LE operations.
- Foundation: the XML parser has been redesigned.
- Bug Fix: SMTP: 5.1c1: Banned IP addresses could not be added.
- Bug Fix: SIP: 5.1c3: symbolic "sent by" Via fields support was broken.
- Bug Fix: RULES: 5.1c3: domain object creating/renaming caused calls to the External AUTH program.
- Bug Fix: RULES: 5.0c3: under certain conditions macro symbol processing could cause an infinite loop.
- Bug Fix: Lawful Intercept: 5.1c2: Authenticated requests in secondary domains were processed incorrectly.
- Bug Fix: WebAdmin: 5.1c1: Account Import operation could crash the server.
- Bug Fix: SIGNALRULES: 5.1c3: the DivertAll rule editor did not show the currently selected options.
- Bug Fix: CLUSTER: 5.0: the CreateDomainPBX operation did not propagate correctly.
- Bug Fix: CLUSTER: 5.0: NATed IP addresses could not be updated in Cluster.
- Bug Fix: Kernel: 4.0: the DIGEST-MD5 parser did not accept spaces after quoted strings.
- Bug Fix: TempFiles: 5.1c1: file write errors could cause loops.
- Bug Fix: CALLLEG: 5.1c2: failure of a refreshing OPTION request did not cause call disconnect.
- Bug Fix: Telnum: 5.1c2: "full dump" in the Main Domain cut the last account name symbols.
- Bug Fix: WebMail: 5.1c2: created Mailboxes were added to the subscription list w/o converting their names into UTF-7.

## 5.1c3 14-Aug-06

Valid Core License Keys: issued between 01-Aug-2004 and 31-Oct-2004, or on or after 01-Aug-2005.

- SMTP: the Reverse-Connect option now checks all available MX relays.
- CLUSTER: GetWebFilesInfo operation is "clusterized".
- CLUSTER: the "relay" SIP Farm option is implemented. Backend Servers now can relay their SIP requests via SIP Farm members.
- CALLLEG: incoming application/dtmf INFO requests are supported now.
- SIGNAL: redirect responses for requests from local sources are processed internally.
- SIGNAL: non-dialog OPTIONS requests directed to applications are processed internally.
- SIGNAL: Microsoft "provisioning" event package is supported now.
- SIGNALRULES: the Discard Rules action is implemented.
- MAPI: the version 1.2.2 of the MAPI Connector is included: Delegation support has been implemented.
- XMPP: Jabber protocol extensions are implemented.
- MEDIAPROXY: proxy management moved from the SIP component to the SIGNAL component.
- PBX: now the StartBridge() operation can be used for pending incoming calls.
- PBX: the StartCall operation now supports the Expires parameter.
- PBX: now the "loop detector" suspends, but does not abort long-running applications.
- PBX: the "CallID Block", "Call Divert" options are implemented.
- RADIUS: the optional RADIUSPassword Account setting is supported now.
- CG/PL: the RADIUSCall operation is implemented.
- CG/PL: the STRING(ipAddressPort) function now encloses the IP address into brackets.
- CG/PL: the RejectBridge() procedure now has 2 parameters.
- CG/PL: ObjectToString and TextToObject function now use the #null# string to represent a null-value.
- XMLAPI: Calendar management is implemented.
- XMLAPI: Contact management is implemented.
- XMLAPI: Incoming call handling is implemented.
- MEDIACHANNEL: virtual timing processing has been changed.
- CLI: the Real-Time Application Control operations are implemented.
- CLI: the GETTEMPCLIENTIPS and GETTEMPBLACKLISTEDIPS commands enclose reported IP addresses into brackets.
- Triggers: the current Server time can be inserted into generated reports.
- Domains: the IP -> Domain mapping has been optimized (for systems with 10,000+ domains).

- Statistics: RADIUS and TFTP elements are implemented.
- SIP: symbolic "sent by" Via fields are supported now.
- Bug Fix: 5.1c1: CLUSTER: Inter-cluster SIP and HTTP addresses were sent incorrectly.
- Bug Fix: 5.1c1: WebAdmin: Account settings update caused PSTN Settings reset.
- Bug Fix: 5.1c1: SIP: URIs with spaces in front of ';' separators were processed incorrectly.
- Bug Fix: 5.1c1: CGPL: the Impersonate function did not work for secondary domains.
- Bug Fix: 5.1c1: SMTP: unaligned buffers could cause crashes on some platforms.
- Bug Fix: 5.1c1: SDP: parser could crash if the media line contained no codec data.
- Bug Fix: 5.1c1: Queue Rules: if an Account had a disabled Rule, the Domain-wide Rules were not applied.
- Bug Fix: 5.1c1: Telnum: deleted Accounts were not removed from the Telnum hash resulting in garbage in telnums.data files.
- Bug Fix: 5.1c2: TLS Major-level logging could crash the server.
- Bug Fix: IMAP: 4.0: certain forms of the FETCH command could crash the server.

**5.1c2** 30-Jun-06
`Valid Core License Keys:` issued between 01-Jun-2004 and 31-Oct-2004, or on or after 01-Jun-2005.

- Admin: Lawful Intercept for Signals is implemented.
- Admin: Debug IP Addresses settings (Network pages) are implemented.
- WSSP: now all string prefixes (HTML, JAVASCRIPT, etc.) support numeric data.
- XIMSS: the Signal Management, Mailbox Management, vocabulary operations are implemented.
- SIP/HTTP: multi-line WWW-Authenticate/Proxy-Authenticate header fields are supported now.
- CG/PL: FINDREGEX, TOUPPERCASE, TOLOWERCASE functions are implemented.
- CG/PL: the EMAILTOSIPURI function now uses the URI escape sequences.
- PBX: the 486 (Busy Here) error code can be used in the rejectCall() operation.
- CLI: SETACCOUNTPASSWORD now accepts the TO keyword again, though it is deprecated (the PASSWORD key should be used).
- Bug Fix: 5.1c1: QUEUERULES: the Each Route/Any Route conditions disappeared from the WebAdmin menus.
- Bug Fix: 5.1c1: PBX: far-end NATed calls were processed incorrectly.
- Bug Fix: 5.1c1: WebUser: the "secure display" option for embedded images did not work.
- Bug Fix: 5.1c1: SignalRules: the StopProcessing action did not really stop rule processing.
- Bug Fix: 5.1c1: SIP: the "via" routing addresses were ignored.
- Bug Fix: 5.1c1: POP: certain mail clients could crash the server when opening empty INBOXes.

**5.1c1** 15-Jun-06
`Valid Core License Keys:` issued between 01-Jun-2004 and 31-Oct-2004, or on or after 01-Jun-2005.

- All components have been modified to support IPv6 network addresses.
- Foundation: all string-keyed dictionaries now use object-type keys.
- Foundation: case-insensitivity comparison is implemented for the basic Cyrillic and Greek symbols.
- XMPP: the XMPP Core protocol is implemented.
- XMLAPI interface is implemented.
- XIMSS: the XIMSS module is implemented.
- BSDLog module is implemented.
- Foundation: regular expression support is implemented.
- Kernel: MIME parser has been redesigned.
- LOG: record format changed: 3 digits for milliseconds, session/packet counters changed from 5 to 6 digits.
- LOG: the Keyed and RegEx options are implemented.
- LOG: sending Log records to remote syslog servers is implemented.
- ROUTER: ENUM-search (RFC2916) is implemented.
- ROUTER: telephone number processing (the "telnum" domain) is implemented.
- ROUTER: now mailbox/application (name#) and detailing (+name) portions are preserved when Reroute Unknown settings are applied.

- Admin: the Telephone Number assignment for Accounts is implemented.
- QUEUE: the "synchronous" Enqueuer mode is implemented (messages rejected with Rules/Filters are rejected on the protocol level).
- QUEUE: the Sensitivity header field is processed (its `private` value sets the Hidden message flag).
- QUEUE: the Suppress Failed Delivery Reports option is added to the Reject functions on the Queue and Message Monitor page.
- DNR: now "resource records" in MX and SRV responses are utilized to avoid additional A-type lookups and to support IPv6 DNS records.
- DOMAIN: renameAccount and removeAccount operations now try to kill all active Account sessions first.
- PBX: sending DTMF via RTP (RFC2833) is implemented for both the direct and bridged modes.
- MAPI: the version 1.2.1 of the MAPI Connector is included: Delegation support has been implemented.
- NETWORK: WAN IPv6 Address setting is implemented.
- FTP: the Use WAN Address option is implemented.
- FTP: now Passive Mode transfers use ports from the TCP Port range specified in Network Settings.
- FTP: RFC2428 (IPv6 and NATs) is implemented.
- FTP: access to other Accounts Sites (via ~account@domain/ prefix) is supported now.
- DIRECTORY: now the Access Right restrictions are applied to the "top" record, too.
- DIRECTORY: non-DN search operations have been optimized.
- WebAdmin: Directory Management has been switched to the WSSP (Skins) Interface.
- WebAdmin: parts of Settings Management have been switched to the WSSP (Skins) Interface.
- WebAdmin: the Domain and Account management page is internationalized (the language setting is taken from the administrator preferences).
- WebAdmin: now Custom Settings can be modified on the Account Defaults pages.
- WebAdmin: Forwarder management has been modified (the "All Forwarders" page has been removed).
- WebAdmin: the DNR settings have been moved to a separate Network Settings page.
- WebAdmin: the Account administration pages have been rearranged.
- WebAdmin: the Domain Object List now displays the number of registered Real-Time devices for each displayed Account.
- WebAdmin: the Signal Info page (current Registrations, Roster, Packages) is implemented.
- WebAdmin: the WebAdmin Layout (Skin selection) setting is implemented.
- MIME: search algorithm has been modified to support multi-charset message header search.
- SIP: the Media Proxy manager now supports the UPDATE operations.
- SIP: RFC3325 (P-Asserted-Identity) is implemented.
- SIP: the NOTIFY requests generated by the Server now include the Contact: field.
- SIP: Windows Messenger/RTC Directory search requests are supported now.
- SIP: NoSubMWI "Workaround" is implemented to support devices that fail to subscribe to MWI (including Cisco phones).
- CLUSTER: the MakeReady/MakeNonReady operations now work for the Active Controller and Frontend Servers.
- CLUSTER: Temp Blacklisted addresses are automatically distributed to all Cluster members now.
- CLI: the SETTEMPBLACKLISTEDIPS command is implemented.
- CLI: the KILLACCOUNTSESSIONS command is implemented.
- CLI: the RENAMEFORWARDER command is implemented.
- CLI: the GETACCOUNTTELNUMS and SETACCOUNTTELNUMS commands are implemented.
- CLI: the REJECTQUEUEMESSAGE now supports the NONDN parameter.
- CG/PL: the APPENDSITEFILE function is implemented.
- CG/PL: the *SITEFILE functions can work with files stored in other Accounts.
- CG/PL: the IMPERSONATE function is documented.
- CG/PL: the ROUTEADDRESS, DIRECTORYSEARCH functions are implemented.
- CG/PL: the HTTPCALL function parameters are extended.
- CG/PL: the Preference management functions are implemented.
- CALLLEG: Session Refresheners are implemented.
- SIGNAL: the Account-level (Account and Domain) Incoming Signal Rules are implemented.
- SIGNAL: the "Real-Time settings" are deprecated.

- SIGNAL: simplified Rules are implemented.
- SIGNAL: the Registered Contacts Limit is implemented.
- SIGNAL: CDR Logs are stored inside the SystemLogs directory now.
- RULES: the time-based conditions in the Account-level Rules now use the Account time zone.
- RULES: the Time of Day condition now supports the "in" and "not in" operations.
- SNMP: the realTimeNode and Foundation elements have been added.
- FreeBusy: the FreeBusy file is now deleted automatically when the default Calendar Mailbox is updated using any protocol/method.
- Security: DIGEST-MD5 authentication supports Impersonation now (the "authzid" parameter).
- WebUser: the new "stock" Skin is implemented. The old "stock" skin is available as the "Classic" one.
- WebUser: the Thai language support has been added.
- WebUser: the Dial settings page has been implemented.
- WSSP: the INCLUDE parameters are implemented.
- WSSP: the ELIF element has been implemented.
- WSSP: the YESNO, CURRENTTIME functions are implemented.
- WSSP: the FORALL/FOREACH element can use both arrays and dictionaries now.
- WSSP: the DAYTIMEMENU and related menus now use the "hourMinute" format element.
- TFTP: access to Account Sites is done on behalf of the "tftpuser" Account now.

# Summary

## Foundation

- All string-keyed dictionaries now use object-type keys.
- All libraries have been modified to support IPv6 network addresses.
- Case-insensitivity comparison is implemented for the basic Cyrillic and Greek symbols.
- The XML parser has been redesigned.
- Regular expression support is implemented.
- The TIS-620 alias for ISO-8859-11 is implemented.
- Case-insensitive "wildcard" search operations support non-Latin symbols.

## Kernel

- Multilingual Guide system is implemented.
- The Russian Guide is included.
- All components have been modified to support IPv6 network addresses.
- Dying async objects now process pending event messages.
- secondary TCP channels now can use Non-Blocking sockets, too.
- multi-line RFC2231 fields are supported now.
- E-mail usernames can now have a leading "!" symbol.
- the {SSHA} password encryption method is supported.
- NorthAmerica/* time zone descriptors are updated to reflect the changes in USA DST.
- Kerberos NT-SMTP-NAME and NT-ENTERPRISE are supported now.
- the DIGEST/NTLM parameter parser modified to accept more white-space separators.
- the "server zone" is renamed from "***" to "HostOS" and turned into a "real" time-zone.
- bracket-comments are rejected in E-mail addresses that are not supposed to have comments (SMTP addresses, certificate E-mails, etc.)
- the Digest "uri" parameter size limit has been increased.

## Platform

- Linux/s390x (64-bit) version is released.
- MacOSX: the startup scripts are moved from the /System/Library to the /Library directory.

- Solaris: IP read buffer has been increased to allow for 1000+ local IPv4 addresses.

## Pronto

- The Pronto! Flash-based XIMSS interface client is included.

## Admin

- the Signal "Service" setting is implemented.
- Initial Installation procedure has changed: the initial postmaster password is set via the WebAdmin Interface.
- The Telephone Number assignment ("Telnums") for Accounts is implemented.
- Debug IP Addresses settings (Network pages) are implemented.
- Lawful Intercept for Signals is implemented.
- The PSTN Settings Domain Access Right is implemented.
- Initial Message text in Account Template can contain macro-symbols.

## Log

- The Open Showing Last option has been moved to Admin Preferences.
- Record format has been changed: 3 digits for milliseconds, session/packet counters changed from 5 to 6 digits.
- The Keyed and RegEx options are implemented.
- Sending Log records to remote syslog servers is implemented.
- Crash-level records can be processed with Trigger Handlers.

## NETWORK

- WAN IPv6 Address setting is implemented.
- UDP Listeners support multiple sockets now.
- "resource records" in MX and SRV DNS responses are utilized to avoid additional A-type lookups and to support IPv6 DNS records.

## CLUSTER

- the heartbeat timeout decreased to 30 seconds.
- Signal: the authenticated/redirector information is passed between Cluster members.
- The "nonce" distribution algorithm has been improved.
- The information about the packet target address is passed within the SIP Farm.
- GetWebFilesInfo operation is "clusterized".
- The "relay" SIP Farm option is implemented. Backend Servers now can relay their SIP requests via SIP Farm members.
- The MakeReady/MakeNonReady operations now work for the Active Controller and Frontend Servers.
- Temp Blacklisted addresses are automatically distributed to all Cluster members now.
- Support for "stateful" UDP load balancers has been added.

## ROUTER

- ENUM-search (RFC2916) is implemented.
- Telephone number processing (the "telnum" domain) is implemented.
- Now mailbox/application (name#) and detailing (+name) portions are preserved when Reroute Unknown settings are applied.
- The Domain-level Access To Unknown settings are implemented.
- Table records can have more than one Mail/Signal/Access prefix.

## MIME

- MIME parser has been redesigned.
- search algorithm has been modified to support multi-charset message header search.
- RFC2231 language/charset tags are processed now.

## Security

- the Domain S/MIME Escrow Certificate support is implemented.
- S/MIME signature certificate authority chains are processed now.
- Multi-line WWW-Authenticate/Proxy-Authenticate header fields are supported now.
- DIGEST-MD5 authentication supports Impersonation now (the "authzid" parameter).
- delays after login failures have been moved "up" to the module components to decrease resource consumption.

## TLS

- now the certificate subjectAltName elements can be used for Certificate-base authentication.
- the SSL 3.0 "no_certificates" Alert message is supported now (Firefox 2.0 may issue it when a client-side certificate is requested).
- the Abort on Wrong Client Certificate option is implemented.
- wildcard certificates are accepted now.

## Domains/Accounts

- the IP -> Domain mapping has been optimized (for systems with 10,000+ domains).
- renameAccount and removeAccount operations now try to kill all active Account sessions first.

## QUEUE

- the "synchronous" Enqueuer mode is implemented (messages rejected with Rules/Filters are rejected on the protocol level).
- the Sensitivity header field is processed (its `private` value sets the Hidden message flag).
- the Suppress Failed Delivery Reports option is added to the Reject functions on the Queue and Message Monitor page.

## XMPP

- The XMPP module is implemented.

## XIMSS

- The XIMSS module is implemented.

## BSDLog

- The BSDLog module is implemented.

## CALLLEG

- PRACK processing changed to avoid cross-wiring with 200-INVITE in B2BUA applications.
- dialogs started with supplementary 2xx-INVITE responses are gracefully terminated now.
- 2xx-INVITE responses without SDP are supported now.
- the StartCall function now supports the P-CGP-Private parameter.
- responses to INVITE with Supported:timer now contain the Require:timer field.
- refresh requests are now sent 1/2 into the refresh interval.
- refresh requests are considered failed on 408 and 481 codes only.
- the uri-parameters (such as user=phone) are now recorded into the dialog context.

on a 422 response, the INVITE request is resent with the returned Min-SE value.

- incoming application/dtmf INFO requests are supported now.
- Session Refresheners are implemented.
- the StartCall parameters are remembered and reused when REFER'ed INVITE requests are composed.
- the Allow/Supported fields are included into 200-INVITE responses.
- deadlocks are detected when both sides of a bridged call try to re-INVITE at the same time.

## PBX

- the StartBridgedCall function is implemented (it is used to implement "transparent bridging" for B2BUA applications).
- the StartBridge() operation can be used for pending incoming calls.
- the StartCall operation now supports the Expires parameter.
- the "loop detector" suspends, but does not abort long-running applications.
- the "CallID Block", "Call Divert" options are implemented.
- the 486 (Busy Here) error code can be used in the rejectCall() operation.
- sending DTMF via RTP (RFC2833) is implemented for both the direct and bridged modes.
- the TransferCall function is documented.
- SendDTMF() operation sends INFO-requests if the call is actually bridged.
- the Alternative Number and the Home PSTN Number settings are implemented.
- the "mailbox service" application now supports voicemail forwarding.
- the Transfer and CallInfo functions are documented.
- bridged provisioning responses are "stacked" to compensate for different egress and ingress signaling speeds.
- re-INVITEs sent after BridgeBreak contain an offer with all server-supported codecs.
- the SetInterruptOnDTMF function is implemented.
- the SetBridgedProvision function is implemented.

## SIGNAL

- the Flow Control options are implemented.
- supplementary 200-INVITE responses are upstreamed now.
- the "Auth all outgoing INVITEs" option is implemented.
- the RTC (Windows Messenger) <-> Roster integration is implemented.
- Request URI parameters are preserved when a request is routed to a remote domain.
- redirect responses for requests from local sources are processed internally.
- non-dialog OPTIONS requests directed to applications are processed internally.
- Microsoft "provisioning" event package is supported now.
- Discard Rules action is implemented.
- "collapse proxy" processing has been modified.
- media proxy management moved from the SIP component to the SIGNAL component.
- the Account-level (Account and Domain) Incoming Signal Rules are implemented.
- the "Real-Time settings" are deprecated.
- simplified Rules are implemented.
- the Registered Contacts Limit is implemented.
- CDR Logs are stored inside the SystemLogs directory now.
- the "must proxy" flags are now set directly in SDP, so they are preserved when a call travels between protocols and tasks.
- the Router supports the special "incomplete" address now. Signals routed to that address are rejected with the 484 Address Incomplete error.

## SIP

- the "Send 100-Trying" options are implemented.
- CANCEL requests are postponed till the first provisioning response.

symbolic "sent by" Via fields are supported now.
- the Media Proxy manager now supports the UPDATE operations.
- RFC3325 (P-Asserted-Identity) is implemented.
- the NOTIFY requests generated by the Server now include the Contact: field.
- Windows Messenger/RTC Directory search requests are supported now.
- NoSubMWI "Workaround" is implemented to support devices that fail to subscribe to MWI (including Cisco phones).
- the TimerB setting is implemented.
- negative-ACK requests contain the same authentication fields as the failed INVITEs.
- PUBLISH requests w/o Expires: header field are accepted now.
- URI and field parameter value escaping methods have been improved.
- a workaround for several gateways and UA: if a 18x response was sent, a locally-generated final response now reuses the same To-tag.
- packets with multipart bodies with an SDP sub-part are supported now.
- the UDP TOS Tag option is implemented.
- incorrectly formed Refer-To fields (Mediatrix) are accepted now.
- the Send P-Asserted-Identity option is implemented.

## RULES

- the time-based conditions in the Account-level Rules now use the Account time zone.
- the Time of Day condition now supports the "in" and "not in" operations.
- the Submit Address condition is now supported in Signal Rules, too.

## DIRECTORY

- binary attributes (such as certificates) in Remote Units are sent with the ";binary" suffix (if the LDAP protocol version is set to 3).
- unauthenticated requests match only Access Rights specified for "anyone" Bind DN.
- the Access Right restrictions are applied to the "top" record, too.
- non-DN search operations have been optimized.
- search operations now support multi-value attributes.
- the stringMatch search operation is redesigned.
- the 'displayName' processing moved to the Directory 'Local Units'.

## Mailbox

- UTF-7 validity of new Mailbox names is checked now.
- the Junk, $Label1, $Label2, $Label3 message flags are available now. In TextMailboxes these new flags can be set only for the messages stored with the latest 5.1 versions.
- the BSD Mailbox "restore damaged counter" procedure is extended to index files.
- locking operations can time-out now. The time-out is set to 30 seconds.
- the number of messages with the Service flag is stored in Mailbox meta-data now.

## File Storage

- .meta-files are implemented.

## HTTP

- the HTTP settings have been moved from the WebUser page to the HTTP module pages.

## HTTPOutput

- if the returned data is a text in a non-UTF8 charset, the returned data is converted into UTF-8.

- "chunked" responses are supported now.

## LDAP

- special processing for the displayName attribute is extended to the GE and LE operations.
- the "anr" pseudo-attribute is supported.

## IMAP

- if an 8-bit symbol is met in an e-mail address part, the FETCH ENVELOPE element now presents that part as a literal.
- the MOVE commands are implemented.
- the ENCRYPTED parameter is documented.

## ACAP

- literals in AUTHENTICATION challenge responses are accepted now.

## MAPI

- the MAPI Connector version 5.1.14 is included.

## SMTP

- an option to start Parallel Channels for the specified queue size is implemented.
- the Reverse-Connect option now checks all available MX relays.
- when non-220 initial prompt is received, the QUIT command is sent before disconnecting.

## SNMP

- the realTimeNode and Foundation elements have been added.

## MediaServer

- codec processing has been changed (to support stateful codecs).
- the A/V socket TOS byte can be set now.
- virtual timing processing has been changed.
- the Source Port Restriction option is implemented.
- the sending addresses are built using "other side Media IP" whenever it is known.

## CALENDAR

- the FreeBusy file is now deleted automatically when the default Calendar Mailbox is updated using any protocol/method.
- now "exception" events publishing does not conflict with the main event publishing.

## CG/PL

- the StoreCDR procedure is implemented.
- the RADIUSCall operation is implemented.
- the STRING(ipAddressPort) function now encloses the IP address into brackets.
- the RejectBridge() procedure now has 2 parameters.
- the ObjectToString and TextToObject function now use the #null# string to represent a null-value.
- the FINDREGEX, TOUPPERCASE, TOLOWERCASE functions are implemented.
- the EMAILTOSIPURI function now uses the URI escape sequences.
- the APPENDSITEFILE function is implemented.

- the *SITEFILE functions can work with files stored in other Accounts.
- the IMPERSONATE function is documented.
- the ROUTEADDRESS, DIRECTORYSEARCH functions are implemented.
- the HTTPCALL function parameters are extended.
- the Preference management functions are implemented.
- the MailboxRedirectByUID, MailboxForwardByUID functions are implemented.
- the ReadGroupMembers function can access Groups in other Domains.
- the RADIUSCall function now supports vendor-specific attributes.
- the READTELNUMS function is implemented.
- the UPDATEACCOUNT[MAIL|SIGNAL]RULE functions are documented.

## RADIUS

- more error conditions result in negative responses instead of request dropping.
- the optional RADIUSPassword Account setting is supported now.
- the Statistics elements are implemented.
- vendor-specific attributes are supported now.
- External Helper now gets all request attributes and an additional Password attribute containing the "shared secret" string.

## FTP

- the Use WAN Address option is implemented.
- Passive Mode transfers use ports from the TCP Port range specified in Network Settings.
- RFC2428 (IPv6 and NATs) is implemented.
- access to other Accounts Sites (via ~account@domain/ prefix) is supported now.
- the Passive Mode setting is implemented.
- file/directory name limit is increased to 1024 bytes.

## TFTP

- if a file name contains a leading slash and no other slash symbols, the leading slash is removed.
- the Statistics elements are implemented.
- access to Account Sites is done on behalf of the "tftpuser" Account now.
- the Try IP-Address Directory option is implemented.
- the Run Sessions on Controller option is implemented.

## SDP

- symbolic names are accepted in the "o" lines.

## CLI

- the Real-Time Application Control operations are implemented.
- the GETTEMPCLIENTIPS and GETTEMPBLACKLISTEDIPS commands enclose reported IP addresses into brackets.
- SETACCOUNTPASSWORD now accepts the TO keyword again, though it is deprecated (the PASSWORD key should be used).
- the SETTEMPBLACKLISTEDIPS command is implemented.
- the KILLACCOUNTSESSIONS command is implemented.
- the RENAMEFORWARDER command is implemented.
- the GETACCOUNTTELNUMS and SETACCOUNTTELNUMS commands are implemented.
- the REJECTQUEUEMESSAGE now supports the NONDN parameter.
- the GETSERVERTRUSTEDCERTS, SETSERVERTRUSTEDCERTS, GETCLUSTERTRUSTEDCERTS, SETCLUSTERTRUSTEDCERTS commands are documented.

- the [GET|SET][SERVER|CLUSTER|DOMAIN|ACCOUNT]RULES commands are renamed into [GET|SET][SERVER|CLUSTER|DOMAIN|ACCOUNT]MAILRULES. Old commands are still supported.
- the [GET|SET][SERVER|CLUSTER|DOMAIN|ACCOUNT]SIGNALRULES commands are documented.
- the UPDATEACCOUNT[MAIL|SIGNAL]RULE commands are documented.
- the UPDATEWEBUSERDEFAULTS command is implemented.

## WebAdmin

- the WebAdmin Interface has been redesigned.
- the WebAdmin Interface has been switched to Skins. The WebAdmin application directory has been removed.
- the Separate WebAdmin Realms option is implemented. By default the WebAdmin Interface now uses one authentication realm.
- the disabled Mobile service is now ignored for the Main Domain only.
- the Interface Type (Expertise Level) concept is implemented.
- the Domain and Account management page is internationalized (the language setting is taken from the administrator preferences).
- now Custom Settings can be modified on the Account Defaults pages.
- Forwarder management has been modified (the "All Forwarders" page has been removed).
- the DNR settings have been moved to a separate Network Settings page.
- the Account administration pages have been rearranged.
- the Domain Object List now displays the number of registered Real-Time devices for each displayed Account.
- the Signal Info page (current Registrations, Roster, Packages) is implemented.
- the WebAdmin Layout (Skin selection) setting is implemented.
- requests to the server-specific pages in a Dynamic Cluster are redirected to the Cluster realm.
- the Access and Service module settings are specified on separate pages now.

## WebUser

- the From columns are replaced with To columns not only for Send and Draft Mailboxes, but for their sub-mailboxes, too.
- if the ReplyHeader Preference is an empty string, the original message text is not added to reply messages.
- when a message with an empty body is being stored as a Draft, the message body part is not omitted, so the Edit Draft operation does not try to edit the first attachment.
- calendaring page processors are completely converted to process data in local-time terms.
- the Language parameters are "cleaned" to disable URL-embedded scripts.
- the Call Log viewer is implemented.
- the Previous Login and the Last Failed Login information is available on the "hello" (Summary) page.
- the new "stock" Skin is implemented. The old "stock" skin is available as the "Classic" one.
- the Dial settings page has been implemented.
- Service messages are not included into the message counter displayed on the "mailboxes" page.
- html-cleanup processes all standard attributes with URI-type values.
- html-cleanup process removes scripting tags together with their contents.
- USENET signatures ("-- " lines) are processed when rendering "format=flowed" messages.
- the Ukrainian, Korean, Thai, Polish languages have been added.
- the number of Mailbox Viewer fields a user can specify is limited now.
- Tasks are sorted by their Due dates, and then - by their modification dates.

## WSSP

- the FOREACHREV element is implemented.
- the INCLUDE parameters are implemented.
- the ELIF element has been implemented.
- the YESNO, CURRENTTIME functions are implemented.
- the FORALL/FOREACH element can use both arrays and dictionaries now.

the DAYTIMEMENU and related menus now use the "hourMinute" format element.

- now all string prefixes (HTML, JAVASCRIPT, etc.) support numeric data.
- the ^ (XOR) operation and the BOOLARRAY() function are implemented.
- the FOREACHINC element is documented.

**Triggers**

- the current Server time can be inserted into generated reports.

**Utils**

- the MoveIMAPMail command now copies all message flags supported with the target server.

---

**CommuniGate Pro**

Introduction
Installation
SysAdmin
Network
Objects
Storage
E-mail
Real-Time
Access
Services
Directory
Clusters
Applications
Miscellaneous
Licensing

WebMail
Pronto!
PBX

| Introduction | Features | History | How To | Help Me |

# Version 5.0 Revision History

- **6.2**
- **6.1**
- **6.0**
- **5.4**
- **5.3**
- **5.2**
- **5.1**
- **5.0: Summary**
- **4.x**
- **3.x**
- **2.x**
- **1.x**

RSS

**5.0.14** 28-Aug-07
`Valid Core License Keys`: issued on or after 1st of Oct'2003.

- Bug Fix: RULES: 5.0.1: malformed Rules could crash the server.
- Bug Fix: HTTP: 4.3: routing requests to an Account Storage subdirectory could cause memory corruption.
- Bug Fix: WebAdmin: 5.0.3: opening the Alerts page for an non-existent account could crash the server.
- Bug Fix: TFTP: 4.3: certain retrieval operations could cause memory corruption.
- Bug Fix: ACAP/IMAP: 4.2: malformed Q-strings could crash the server.

**5.0.13** 25-Nov-06
`Valid Core License Keys`: issued on or after 1st of Oct'2003.

- EVENT: the SUSBCRIBE requests with the same CallID and From-tag clean the existing matching subscriptions.
- Bug Fix: WebUser: 5.0c1: if the login page was opened using the wml/imode/imodejp references, the response Content-Type was not set.
- Bug Fix: CLI: 4.3c3: the FindWebUserSession command could delay the WebUser Session timeouter.
- Bug Fix: Domain: 4.3: the number of domain forwarders could be calculated incorrectly after restart.
- Bug Fix: WSSP: 5.0c2: when HTMLSUBSTMAILBOX function was used with non-UTF8 pages, Mailbox name charset could be converted twice.
- Bug Fix: ROUTER: 5.0c3: All-Local Routing Records could leave the secondary domain name intact, causing Router loops.
- Bug Fix: CLUSTER: 4.0: recovery after detecting an over-the-limit inter-cluster message caused synching problems.

**5.0.12** 14-Oct-06
`Valid Core License Keys`: issued on or after 1st of Oct'2003.

- WebUser: the Language parameters are "cleaned" to disable URL-embedded scripts.
- Bug Fix: CG/PL: 5.0c5: readGroup() with a non-string parameter could crash the server.
- Bug Fix: CLUSTER: 4.1: OS/2 or Windows-based frontends could corrupt messages during inter-cluster mail delivery.
- Bug Fix: SIP: 4.2: corrupted TCP messages could crash the server.
- Bug Fix: CLUSTER: 5.0c2: PBX environment management had file transfer problems.

**5.0.11** 11-Sep-06
`Valid Core License Keys:` issued on or after 1st of Oct'2003.

- License Keys issued after 01-Sep-2006 are accepted now.
- Bug Fix: RULES: 5.0c3: under certain conditions macro symbol processing could cause an infinite loop.
- Bug Fix: CLUSTER: 5.0: the CreateDomainPBX operation did not propagate correctly.
- Bug Fix: Kernel: 4.0: the DIGEST-MD5 parser did not accept spaces after quoted strings.

**5.0.10** 11-Aug-06
`Valid Core License Keys:` issued on or after 1st of Oct'2003.

- MAPI: the version 1.1.37 of the MAPI Connector is included.
- CALENDAR: Apple iCAL items with fictitious time zones are accepted now.
- Bug Fix: Kernel: 4.2: the Digest-MD5 qop and ncount parameters were composed as quoted-strings.
- Bug Fix: Cluster: 4.2: /login/ authentication did not pass the original URL domain name to backends.
- Bug Fix: IMAP: 4.0: incorrectly used extension command could crash the server.
- Bug Fix: IMAP: 4.0: certain forms of the FETCH command could crash the server.

**5.0.9** 26-Mar-06
`Valid Core License Keys:` issued on or after 1st of Oct'2003.

- MAPI: the version 1.1.36 of the MAPI Connector is included.
- CG/PL: the RemoveElement and InsertElement procedures have been added.
- SIP: unexpected "rport" branch-parameters are ignored now.
- SIP: URI/Routes containing secondary Domain names can be processed as "self-routes" now.
- SIP: "route by realm" is restricted to NTLM methods only.
- LDAP: search operations for the displayName attribute now check the UID and CN attributes.
- Router: Unknown Domain names can be discarded or rerouted even if they contain symbols illegal for Domain objects.
- Bug Fix: Calendar: 5.0.7: the TZID property was stored twice (once - with an empty value).
- Bug Fix: CLI: 5.0c5: the GETACCOUNTINFO command with an array argument did not accept spaces before the argument.
- Bug Fix: TFTP: 5.0c2: accounts in secondary Domains could not be accessed.
- Bug Fix: IMAP: 3.5: Search operations with ISO-2022 parameters could fail to find certain messages.
- Bug Fix: WebAdmin: 5.0c3: Domain administrator could not use custom domain Skins.

**5.0.8** 03-Feb-06
`Valid Core License Keys:` issued on or after 1st of Oct'2003.

- Domains: Hash processing for very large (over 5,000,000 accounts) domains has been improved.
- Directory: DN processing changed to avoid problems with DNs containing too many (>100) elements.
- Network: the Round-Robin Allocation option is implemented.
- Bug Fix: SIGNAL: 5.0c2: account without registrations could interrupt processing even when it was not the original AOR (but a Group member).

- Bug Fix: SIGNAL: 5.0.1: the Reject With RULE operation did not immediately stop request processing.
- Bug Fix: DOMAIN: 5.0.7: Default WebUser settings were not re-read after a restart.

**5.0.7** 27-Jan-06
`Valid Core License Keys`: issued on or after 1st of Oct'2003.

- MEDIA: DTMF sending stack is implemented.
- SIP: Authentication and Impersonation are separated now.
- MAPI: the version 1.1.33 of the MAPI Connector is included.
- Bug Fix: WebAdmin: 5.0: the Server-wide/Cluster-wide Domain Foldering options could not be set.
- Bug Fix: Account: 5.0c1: the "Max Number of failed Logins" setting did not work.
- Bug Fix: EXTCDR: 5.0: the server expected an additional symbol after the "OK" response.
- Bug Fix: CALENDAR: 5.03: storing TZID parameters unquoted caused problems when a TimeZone name contained "non-safe" symbols (such as ",").
- Bug Fix: PBX: 5.0.5: the 491 response for re-INVITE calls could crash the server.
- Bug Fix: WebUser: 5.0.4: the Decrypt operation did not work on Unix systems.
- Bug Fix: Domains: 5.0: when Directory-based Domains were created, "dc" RDN attributes were incorrectly added to the record data.
- Bug Fix: Rules: 4.3: Domain-wide Rules were stored unsorted.
- Bug Fix: Foundation: 3.0: Negative BER lengths were processed incorrectly.
- Bug Fix: SDP: 5.0.5: the "ptime" attribute was composed incorrectly.

**5.0.6** 30-Dec-05
`Valid Core License Keys`: issued on or after 1st of Oct'2003.

- SIP: added support for "upstreamed" responses for NAT'ed requests lacking the "rport" field.
- PBX: REFER modes supported.
- Bug Fix: SIP: 5.0.5: the MakeCall function could crash if the callers phone misforms the REFER-NOTIFY.
- Bug Fix: PIPE: 4.0: malformed Return-Path header fields could copy garbage data into Submitted messages.

**5.0.5** 23-Dec-05
`Valid Core License Keys`: issued on or after 1st of Oct'2003.

- DNR: the request and response domain name length limits have been increased.
- MAILBOX: case-insensitive search is now used for all meta-info requests on OSes using case-insensitive file systems.
- MediaProxy: non-standard fax (image) media channels are supported now.
- Presence: `CPIM-PIDF` format is supported now.
- HTTP: HTTP_X_FORWARDED_FOR header field value is added to the set of standard CGI environment variables.
- Bug Fix: MediaChannel: 5.0c4: incorrect timer processing could case audio data corruption.
- Bug Fix: PBX: 5.0: CANCEL's for incoming INVITEs were processed incorrectly.
- Bug Fix: MediaProxy: 5.0c3: the proxy state was not completely restored after a failed re-INVITE.
- Bug Fix: SIP: 4.3c5: various malformed packets could crash the parser.
- Bug Fix: SIP: 5.0c1: the "rport" VIA parameter value was not use for response sending.

**5.0.4** 12-Dec-05
`Valid Core License Keys`: issued on or after 1st of Oct'2003.

- PBX: the ProvisionCall() function parameters have been changed.
- PBX: bridged clients now can use codecs not supported by the Media Channel.
- FTP: RFC4217 is implemented (FTP over TLS).
- CLUSTER: "Authenticated Domain Users/Authenticated Administrator" restrictions for "Delivery to All" operation are supported now.
- MAPI: the version 1.1.32 of the MAPI Connector is included (Encrypt/Decrypt commands).

- Signal: the Redirect to and Fork to Signal Rule actions are implemented.
- Bug Fix: PBX: 5.0c3: failure to process SDP for bridged legs could cause crashes.
- Bug Fix: LDAP: 5.0.2: sorting using the displayName attribute did not use the CN attribute.
- Bug Fix: SIGNAL: 5.0c3: relaying to addresses starting with an asterisk could crash the server.
- Bug Fix: Kernel: 5.0c1: Text->HTML conversion created CR-CR-LF sequences on Unix platforms.
- Bug Fix: SMTP: 5.0c1: the "When Receiving" option caused rejects for 'all@domain' addresses.
- Bug Fix: PBX: 5.0: the "service" application did not support the *65 ("clear conference") function.
- Bug Fix: Calendar: 4.3: Recurrences with non-last UNTIL property were processed incorrectly.

**5.0.3** 01-Dec-05
`Valid Core License Keys:` issued on or after 1st of Oct'2003.

- Signal: now SDP recomposition preserves codec priorities.
- Signal: NOTIFY(keep-alive) requests are supported now.
- SIP: the NeedsEpid workaround is implemented now.
- Calendar: now the TZID parameter is stored without quotation marks.
- PBX: voicemails are sent with empty Return-Paths now, so Vacation Rules do not sent autoreplies for voicemails.
- Media: the log record formats have been changed.
- Bug Fix: WebAdmin: 5.0c3: the Account Template page did not contain the "Basic" attributes.
- Bug Fix: WebAdmin: 5.0c1: certain "size-type" settings interpreted the "1G" value as "1".
- Bug Fix: WebAdmin: 5.0: array-type settings on the Advanced Real-Time Settings page were reset to defaults when regular Real-time Settings were updated.
- Bug Fix: FTP: 5.0.2: MKD/RMD operations failed reporting a syntax error.
- Bug Fix: WebAdmin/WebUser: 5.0c2: the "WorkDays" setting processing changed to make it compatible with PBX settings.
- Bug Fix: SIP: the "c" header field was not recognized.

**5.0.3** 16-Nov-05
`Valid Core License Keys:` issued on or after 1st of Oct'2003.

- CG/PL: the LISTSITEFILES function is implemented.
- MAPI: the version 1.1.31 of the MAPI Connector is included.
- FTP: now the MKD, XMKD, LIST, RMD, XRMD commands accept directory paths with trailing slashes.
- FTP: ALERT-closing of an TLS channel is supported now.
- SIP: Proxy Signal/Media Gateway setting is implemented.
- Signal: Redirect settings now accept multiple destinations (comma-separated).
- Signal: Presence from Microsoft clients now expires with the client registration.
- CG/PL: the MailboxExpunge function is implemented.
- LDAP: the displayName function now uses the UID attribute if the CN attribute is absent.
- Bug Fix: SIP: 5.0c6: an incorrectly formed response SDP could cause memory corruption.
- Bug Fix: SIP: 5.0c2: locally generated responses sent upstream did not contain the dialog ID fields.
- Bug Fix: SIP: DTMF INFO requests with codes larger than 10 were processed incorrectly.
- Bug Fix: Account: 5.0c3: MSN password authentication attempts could cause random crashes.
- Bug Fix: PBX/SIP: 5.0c1: inside-the-server bridging could be misinterpreted as a call from a NAT'ed client.
- Bug Fix: SMTP: 5.0c2: non-SMTP (LIST, etc.) Return-Path addresses were treated as SMTP addresses.

**5.0.2** 04-Nov-05
`Valid Core License Keys:` issued on or after 1st of Oct'2003.

- SMTP: the Reverse-Connection option can be set to Add Header now.
- MAPI: the version 1.1.29 of the MAPI Connector is included.
- Bug Fix: PBX: 5.0c1: the AcceptBridge operation did not properly proxy LAN and WAN legs.
- Bug Fix: WebAdmin: 5.0c2: the Show Account Info preference option was processed incorrectly.

- Bug Fix: WebAdmin: 5.0c2: Vacation Message settings could not be updated by a Server Administrator.
- Bug Fix: WebAdmin: 5.0c2: adding Server/Cluster-wide Trusted Certificates could crash SSL/TLS WebAdmin page processor.
- Bug Fix: SIP: 5.0c2: Strict-Route's were processed incorrectly.
- Bug Fix: CLUSTER: slave servers did not process Directory Integration settings updates.

### 5.0 30-Oct-05
`Valid Core License Keys`: issued on or after 1st of Oct'2003.

- Lawful Intercept: the Partial option now reports all message UIDs listed in WebUser "Mailbox views".
- Bug Fix: Signal: 5.0c6: forking (including forking to a Group) could clear the "canRelay" Signal tag.
- Bug Fix: WebAdmin: 4.3: the Set S/MIME Certificate operation actually set the TLS Certificate.
- Bug Fix: SMTP: 4.3: rejecting recipients during heavy-load multi-channel single-host delivery could cause queue structure corruptions.

### 5.0c7 26-Oct-05
`Valid Core License Keys`: issued on or after 1st of Oct'2003.

- MediaChannel: several clients calling from behind the same far-end NAT can participate in a conference now.
- Mailbox: Group Access rights are implemented.
- WebAdmin: the Advanced Real-Time settings can be used in Account Default settings.
- Account: the list of the latest incoming calls is stored in the "account info" data.
- PBX: the ForkCall function is implemented.
- PBX: conferencing with far-end NAT clients is implemented.
- MakeCall: now the "extension part" of a telephone number string is cut off when the number is converted into an URI.
- CGPL: the GETACCOUNTINFO function is implemented.
- CLI: the GETACCOUNTINFO command syntax is extended.
- SIP: the Server does not add Record-Routes to negative and 100 responses now.
- LIST: the "listserver" command processor now accepts list names in the list@domain form.
- LDAP: the "Ignore objectCategory filters" option is implemented.
- LDAP: the "displayName" attribute is generated on-the-fly if needed.
- CALENDAR: now tasks and events with zero durations are processed differently than items with unspecified durations.
- DNR: RFC2915 (NAPTR) is implemented.
- WebUser: S/MIME export format has been modified to allow "Thunderbird/Firefox" to import exported keys without crashing.
- WSSP: EOL symbols inside expressions are processed as "white spaces" now.
- Bug Fix: SIP: 5.0c6: "Relay Restriction" incorrectly processed the request "canRelay" attribute.
- Bug Fix: SMTP: 5.0c4: the ESMTP keyword was not detected in remote server prompts.
- Bug Fix: WebAdmin: 5.0c4: some WebUser (incl. RealTime) string settings could be reset to defaults when they were not listed on the page.
- Bug Fix: WSSP: 5.0c1: the EQUALS(x and y) function was processed as the EQUALSNOCASE() function.
- Bug Fix: HTTP: 4.3: GSSAPI Authentication could fail at random.
- Bug Fix: Domains: 4.0: setAccessRights CLI operations could produce race conditions under a heavy concurrent load.

### 5.0c6 21-Oct-05
`Valid Core License Keys`: issued on or after 1st of Oct'2003.

- SNMP: syntax element names changed from COUNTER to COUNTER64.
- SIP: the Relay restriction setting is implemented.
- SIP: the External Gateway names are separated from the Domain names now (to support multiple "gateway accounts" in the same gateway domain).

Node: now the Node (PBX, etc.) Contacts contain the server IP address.
- WebMail: Explicit login pages for WML and IMode devices are implemented.
- Signal: the Authenticated Rule Condition has been implemented.
- Bug Fix: WebAdmin: 5.0c5: the Access Rights page was not working in secondary domains.
- Bug Fix: SMTP: 5.0c5: Reverse SMTP Check was used for connections coming from Client IP Addresses.
- Bug Fix: SMTP: 5.0c1: the MSN Login method did not work for clients sending LM-only passwords.
- Bug Fix: NODE: 5.0c2: the "Make Call" function could damage server memory.
- Bug Fix: Domains: 4.3: Alias processing could fail on 64-bit Itanium-based systems.

### 5.0c5 14-Oct-05
`Valid Core License Keys`: issued on or after 1st of Oct'2003.

- Signal: calls to `*NN` addresses are routed back to the caller now ("Service" calls).
- PBX: call transfer (REFER, INVITE/Replace) is supported now.
- PBX: the basic PBX services are implemented (see the new PBX section of the manual).
- Security: the Request Client Certificate Domain Setting is implemented.
- Security: the Certificate Authentication Account Setting is implemented.
- Security: the Certificate Authentication method is supported in all session-based protocols and in the WebUser Interface.
- SMTP: connections to the port 587 are not rejected when the Reserved for Clients connection limit is reached.
- SMTP: the Reverse-Connection option is implemented.
- SMTP: the Force AUTH option has been moved to the Domain Settings.
- SMTP: the Verify Return-Path and HELO options are separated now.
- Mailbox: the "service" messages concept is implemented; "service" messages cannot be seen with IMAP/POP clients.
- Router: the nn-mm range format for "matched substring length" is implemented.
- CGPL: the IP Address objects are implemented.
- CGPL/PBX: the RemoteIPAddress function is implemented.
- MAPI: the version 1.1.28 of the MAPI Connector is included.
- Bug Fix: SIP: 5.0c1: External Gateway processors always "fixed" contact addresses.

### 5.0c4 07-Oct-05
`Valid Core License Keys`: issued on or after 1st of Oct'2003.

- Platform: the MacOSX/Intel version is released.
- Router: new wildcard and Escape mechanisms are implemented.
- Signal: the CDR recording and CDR helpers are implemented.
- SMTP, POP, PWD: parsers now accept addresses containing quoted space symbols.
- SIP: the Gateway "Substitute To" setting is implemented.
- SIP: the Gateway "Call Authenticate" setting is implemented.
- FTP: the APPE command is implemented.
- WebUser: now HTML attachments are not "cleaned" when they are copied to the personal File Site.
- WebUser: the Hello.wssp page with account "summary" information has been implemented.
- Router: External Helper Routing is implemented.
- Listener: the Reserve for Clients option is implemented.
- MAPI: the version 1.1.27 of the MAPI Connector is included.
- Bug Fix: SIP: 5.0c3: some log operations could result in server crashes.
- Bug Fix: Router: 5.0c2: All-Domain Aliases were processed incorrectly.
- Bug Fix: Kernel: 5.0c2: Digest-MD5 parameter parsing could enter an infinite loop.
- Bug Fix: SMTP: 5.0c1: the When Receiving Domain Setting did not work properly in secondary Domains.

### 5.0c3 26-Sep-05
`Valid Core License Keys`: issued on or after 1st of Sep'2003.

Router: All-Local Router records are implemented.

- WebAdmin: the Domain & Account administration pages have been changed to use the WSSP interface.
- CLI: the WebAdmin Interface management commands have been removed.
- CLI: the READSUBSCRIBERS command has been documented.
- CLUSTER: now the core SIP infrastructure supports the Dynamic Cluster environment.
- SIP: relaying from a far-end NAT to the same far-end NAT is detected and properly processed now.
- SIP: the module settings have been moved to 4 separated WebAdmin pages.
- SIP: multiple Gateways are supported now.
- SIP: the Gateway Auth Name optional setting is implemented.
- SIP: the Workaround features have been implemented.
- SIP: support for the MacOSX version of Microsoft Messenger is implemented.
- QUEUERULES: the Execute action now supports the [ROUTE] prefix.
- WSSP: the DUMP prefix is implemented (to simplify WSSP code debugging).
- WSSP: the URLSUBST prefix is implemented.
- SMTP: the maximum Keep Trying time has been increased to 3 months.
- MAPI: the version 1.1.26 of the MAPI Connector is included.
- Bug Fix: Domains: 4.3: creating Forwarders and Aliases in freshly-created Domains could fail.
- Bug Fix: Cluster: 4.3: Enabling Keep-Alive could cause WebAdmin authentication problems.
- Bug Fix: Admin: 2.0: concurrent updates of Account Access Rights in the same Domain could cause crashes.
- Bug Fix: Kernel: 5.0c1: UDP receiving did not work properly on some platforms.
- Bug Fix: Mailboxes: 5.0c1: message counters were not cleared before an Index file was read.

**5.0c2** 31-Aug-05

`Valid Core License Keys:` issued on or after 1st of Aug'2003.

- SIP: Record-Route field generation optimized and streamlined.
- Router: the All-Domain Aliases can contain wildcards now.
- PKI: the built-in, Server-wide, Cluster-wide, and Domain-wide Trusted Certificates are implemented.
- SMTP: when using High Security for message sending, the remote server Certificate validity is verified.
- WebUser: validity of Certificates in digitally signed messages is displayed.
- RealTime: the Maximal Registration time period option is implemented.
- Media: DTMF events detection has been modified to compensate for implementation bugs in several popular devices.
- WebApp: the "media" elements have been added to datasets generated with the generic Mailbox and Message components.
- WebUser: the XChange Skin has been added.
- WebUser: new Contact items (vCard) are stored with a UID-based Message-ID header field.
- WebUser: the Remove File(s) (remove message attachments) operation is implemented.
- WebUser: now Bcc addresses are stored with "sent message" copies.
- WebUser: Calendar Recurrence editor now contains the End By control.
- Rules: the SendURL and SendIM actions are implemented.
- CGPL: the HTTPCall, SendEMail, and SendInstantMessage functions are implemented.
- CGPL: the Meeting and StartCall operations have been changed.
- LIST: size settings now support 64-bit values.
- Events: the Send Instant Message and Send URL Notification methods are implemented.
- Bug Fix: Mailbox: 5.0c1: all Text-Mailbox indexes were seen as damaged and they were not used.
- Bug Fix: WebUser: 5.0c1: Mailbox-based and directory-based AddressBooks could not be opened.
- Bug Fix: Kernel: 4.0: incorrect "US/Hawaii" timezone info removed, the -1000 zone renamed into Hawaii/Tahiti.

**5.0c1** 07-Aug-05

`Valid Core License Keys:` issued on or after 1st of Aug'2003.

- Kernel: the CG/PL language has been implemented.
- PBX: the PBX Application environment has been implemented.

- CLI: the PBX Application Administration commands are implemented.
- SIGNAL: the signaling component has been re-implemented.
- SIGNAL: server-wide/cluster-wide Signal Rules are implemented.
- SIP: CANCEL processing has been changed to better support Microsoft packet signing.
- Mailbox: Mailbox formats have been modified to provide more flags per message.
- Mailbox: the Media message flag is implemented.
- IMAP: support for $Media message flag is implemented.
- QueueRules: the Mark operation can now set the Media message flag.
- Mailbox: the total number of unread media messages in maintained.
- SIGNAL: changes in the total number of the unread media messages in INBOX Mailboxes result in "message-summary" data publish operations (support for MWI phone indicators).
- WebAdmin: the PBX Applications editor is implemented.
- WebAdmin: the RealTime Settings pages are implemented.
- WebAdmin: the RealTime Account settings pages are implemented.
- WebAdmin: RealTime and SIP monitor pages are implemented.
- WebAdmin: Inactive addresses (From, To, Cc, etc.) are displayed on the Queue Message Monitor pages now.
- WebSite: the Rename operation is implemented in both WebUser and WebAdmin Interfaces.
- HTTP: the If-Modified-Since header field is supported now.
- WebUser: the DataSet Address Books settings are implemented.
- WebUser: support for audio message parts (MIME and UUEncode) is implemented.
- Admin: Domain Enabled Services settings can be "defaulted" now.
- Admin: the Login Method settings were moved from Obscure settings to Domain settings.
- SMTP: the When Receiving Domain setting is implemented.
- Security: Digest "cnonce" values containing ":" symbols are accepted now.
- Log: now the Log Manager does not create more than 1 file in one minute, even if the current log file has exceeded the size limit.
- MAPI: the version 1.1.25 of the MAPI Connector is included.

---

# Summary

**Platform**

- The MacOSX/Intel version is released.

**Kernel**

- The CG/PL language has been implemented.
- SMTP, POP, PWD: now parsers accept addresses containing quoted space symbols.
- Listener: the Reserve for Clients option is implemented.
- Log: now the Log Manager does not create more than 1 file in one minute, even if the current log file has exceeded the size limit.

**CLUSTER**

- The core SIP infrastructure supports the Dynamic Cluster environment now ("SIP Farm").

**PBX**

- The PBX Application environment has been implemented.

**ROUTER**

- New wildcard and Escape mechanisms are implemented.

- External Helper Routing is implemented.
- All-Local Router records are implemented.
- All-Domain Aliases can contain wildcards now.

## SIGNAL

- The signaling component has been re-implemented.
- Server-wide/cluster-wide Signal Rules are implemented.
- The CDR recording and CDR helpers are implemented.
- The Maximal Registration time period option is implemented.
- The "extension part" of a MakeCall telephone number string is cut off when the number is converted into an URI.
- Calls to *NN addresses are routed back to the caller now ("Service" calls).
- The list of the latest incoming calls is stored in the "account info" data.
- Presence from Microsoft clients now expires with the client registration.
- SDP recomposition preserves codec priorities.
- NOTIFY(keep-alive) requests are supported now.

## RULES

- The Execute action now supports the [ROUTE] prefix.
- The Mark operation can now set the Media message flag.
- The SendURL and SendIM actions are implemented.
- The Authenticated condition has been implemented.

## Domains/Accounts

- Domain Enabled Services settings can be "defaulted" now.
- The Login Method settings were moved from Obscure settings to Domain settings.

## Mailbox

- Group Access rights are implemented.
- Mailbox formats have been modified to provide more flags per message.
- The "service" messages concept is implemented; "service" messages cannot be seen with IMAP/POP clients.
- The Media message flag is implemented.
- The total number of unread media messages in maintained.
- Changes in the total number of the unread media messages in INBOX Mailboxes result in "message-summary" data publish operations (support for MWI phone indicators).

## Security

- The Certificate Authentication method is supported in all session-based protocols and in the WebUser Interface.
- The Request Client Certificate Domain Setting is implemented.
- The Certificate Authentication Account Setting is implemented.
- The built-in, Server-wide, Cluster-wide, and Domain-wide Trusted Certificates are implemented.
- Digest "cnonce" values containing ":" symbols are accepted now.
- The Partial Lawful Intercept option now reports all message UIDs listed in WebUser "Mailbox views".

## FTP

- The APPE command is implemented.
- The MKD, XMKD, LIST, RMD, XRMD commands accept directory paths with trailing slashes.
- ALERT-closing of an TLS channel is supported now.

## HTTP

- The If-Modified-Since header field is supported now.

## LDAP

- The "Ignore objectCategory filters" option is implemented.
- The "displayName" attribute is generated on-the-fly if needed.

## LIST

- The "listserver" command processor now accepts list names in the list@domain form.
- Size settings now support 64-bit values.

## IMAP

- Support for $Media and $Service message flags is implemented.

## MAPI

- The version 1.1.31 of the MAPI Connector is included.

## SIP

- CANCEL processing has been changed to better support Microsoft packet signing.
- Relaying from a far-end NAT to the same far-end NAT is detected and properly processed now.
- The module settings have been moved to 4 separated WebAdmin pages.
- Multiple Gateways are supported now.
- The Gateway Auth Name optional setting is implemented.
- The Workaround features have been implemented.
- Support for the MacOSX version of Microsoft Messenger is implemented.
- The Gateway "Substitute To" setting is implemented.
- The Gateway "Call Authenticate" setting is implemented.
- Record-Route field generation optimized and streamlined.
- The Relay restriction setting is implemented.
- The External Gateway names are separated from the Domain names now (to support multiple "gateway accounts" in the same gateway domain).
- Proxy Signal/Media Gateway setting is implemented.

## SMTP

- The When Receiving Domain setting is implemented.
- Connections to the port 587 are not rejected when the Reserved for Clients connection limit is reached.
- The Reverse-Connection option is implemented.
- The Force AUTH option has been moved to the Domain Settings.
- The Verify Return-Path and HELO options are separated now.
- When using High Security for message sending, the remote server Certificate validity is verified.
- The maximum Keep Trying time has been increased to 3 months.

## SNMP

- Syntax element names changed from COUNTER to COUNTER64.

## Media

- The Media Channel (Media Server) component has been implemented.

## CALENDAR

- Tasks and events with zero durations are processed differently than items with unspecified durations.
- The TZID parameter is stored without quotation marks.

## DNR

- RFC2915 (NAPTR) is implemented.

## Events

- The Send Instant Message and Send URL Notification methods are implemented.

## CLI

- The PBX Application Administration commands are implemented.
- The GETACCOUNTINFO command syntax is extended.
- The WebAdmin Interface management commands have been removed.
- The READSUBSCRIBERS command has been documented.

## WebAdmin

- The Domain & Account administration pages have been changed to use the WSSP interface.
- The PBX Applications editor is implemented.
- The RealTime Signals Settings pages are implemented.
- The RealTime Signals Account settings pages are implemented.
- The RealTime Signals and SIP monitor pages are implemented.
- Inactive addresses (From, To, Cc, etc.) are displayed on the Queue Message Monitor pages now.
- The Advanced Real-Time settings can be used in Account Default settings.

## WebUser

- The DataSet Address Books settings are implemented.
- Support for audio message parts (MIME and UUEncode) is implemented.
- The XChange Skin has been added.
- New Contact items (vCard) are stored with a UID-based Message-ID header field.
- The Remove File(s) (remove message attachments) operation is implemented.
- Now Bcc addresses are stored with "sent message" copies.
- Calendar Recurrence editor now contains the End By control.
- Validity of Certificates in digitally signed messages is displayed.
- Now HTML attachments are not "cleaned" when they are copied to the personal File Site.
- The Hello.wssp page with account "summary" information has been implemented.
- Explicit login pages for WML and IMode devices are implemented.
- S/MIME export format has been modified to allow "Thunderbird/Firefox" to import exported keys without crashing.
- The "media" elements have been added to datasets generated with the generic Mailbox and Message components.

## WSSP

- The DUMP prefix is implemented (to simplify WSSP code debugging).
- The URLSUBST prefix is implemented.
- EOL symbols inside expressions are processed as "white spaces" now.

**WebSite**

- The Rename operation is implemented in both WebUser and WebAdmin Interfaces.

| Introduction | Installation | SysAdmin | Network | Objects | Storage | E-mail | Real-Time | Access | Services | Directory | Clusters | Applications | Miscellaneous | Licensing | WebMail | Pronto! | PBX |
|---|

# Version 4.3 Revision History

- 6.2
- 6.1
- 6.0
- 5.4
- 5.3
- 5.2
- 5.1
- 5.0
- **4.3: Summary**
- 4.2
- 4.1
- 4.0
- 3.x
- 2.x
- 1.x

RSS

**4.3.12** 14-Oct-06
`Valid Core License Keys:` issued on or after 1st of May'2003.

- License Keys issued after 01-Sep-2006 are accepted now.
- WebUser: the Language parameters are "cleaned" to disable URL-embedded scripts.
- Bug Fix: Kernel: 4.0: the DIGEST-MD5 parser did not accept spaces after quoted strings.
- Bug Fix: CLUSTER: 4.1: OS/2 or Windows-based frontends could corrupt messages during inter-cluster mail delivery.
- Bug Fix: SIP: 4.2: corrupted TCP messages could crash the server.

**4.3.11** 11-Aug-06
`Valid Core License Keys:` issued on or after 1st of May'2003.

- CALENDAR: Apple iCAL items with fictitious time zones are accepted now.
- Bug Fix: Kernel: 4.2: the Digest-MD5 qop and ncount parameters were composed as quoted-strings.
- Bug Fix: Cluster: 4.2: /login/ authentication did not pass the original URL domain name to backends.
- Bug Fix: IMAP: 4.0: incorrectly used extension command could crash the server.
- Bug Fix: IMAP: 4.0: certain forms of the FETCH command could crash the server.

**4.3.10** 19-Mar-06
`Valid Core License Keys:` issued on or after 1st of May'2003.

- Directory: DN processing changed to avoid problems with DNs containing too many (>100) elements.
- Bug Fix: Foundation: 3.0: Negative BER lengths were processed incorrectly.
- Bug Fix: IMAP: 3.5: Search operations with ISO-2022 parameters could fail to find certain messages.

**4.3.9** 29-Oct-05
`Valid Core License Keys:` issued on or after 1st of May'2003.

- Bug Fix: Domains: 4.3: Alias processing could fail on 64-bit Itanium-based systems.
- Bug Fix: Domains: 4.3: creating Forwarders and Aliases in freshly-created Domains could fail.
- Bug Fix: Domains: 4.0: setAccessRights CLI operations could produce race conditions under a heavy concurrent load.
- Bug Fix: WebAdmin: 4.3: the Set S/MIME Certificate operation actually set the TLS Certificate.
- Bug Fix: HTTP: 4.3: GSSAPI Authentication could fail at random.
- Bug Fix: SMTP: 4.3: rejecting recipients during heavy-load multi-channel single-host delivery could cause queue structure corruptions.

**4.3.8** 05-Sep-05
`Valid Core License Keys:` issued on or after 1st of May'2003.

- SMTP: the maximum Keep Trying time has been increased to 3 months.
- LIST: list processing changed to ensure that all lists are processed even when the number of LIST processors is insufficient.
- Kernel: incorrect "US/Hawaii" timezone info removed, the -1000 zone renamed into Hawaii/Tahiti.
- Bug Fix: Cluster: 4.3: Enabling Keep-Alive could cause WebAdmin authentication problems.
- Bug Fix: Admin: 2.0: concurrent updates of Account Access Rights in the same Domain could cause crashes.

**4.3.7** 06-Aug-05
`Valid Core License Keys:` issued on or after 1st of May'2003.

- CLUSTER: backend servers mark themselves as "non-ready" before they start the shutdown sequence.
- Calendar: VTODO without a DTSTART date are accepted now.
- Bug Fix: WebUser: 4.1: binary zero data entered into Compose fields could case internal loops.
- Bug Fix: WebUser: 4.1: Tasks starting earlier than the displayed part of the Task page could be displayed incorrectly.
- Bug Fix: CLUSTER: 4.0: Importing Mailing Lists through non-controller servers caused cluster channel de-synching.
- Bug Fix: RPOP: 3.0: malformed RPOP account parameters could cause memory corruption and server crashes.
- Bug Fix: DOMAIN: 4.2: updating Directory-based Domains with an empty IP Addresses list resulted in a WebAdmin error message.

**4.3.6** 02-Jul-05
`Valid Core License Keys:` issued on or after 1st of May'2003.

- TempFile: the SubmitAddress/Domain envelope data is always recorded now (for all submit sources).
- LOCAL: the number of delivery threads can be set to 500 now.
- Lawful Intercept: the From: header field now always contains the full name of the monitored account.
- WebUser: the skinNames dataset element has been added to the login.wssp set. The stock Login.wssp page has the Layout pop-up menu now.
- WebUser: the Hebrew dictionary file has been added.
- Bug Fix: DNR: 2.0: responses with Q>0 and other fields being empty were processed as incorrect responses.
- Bug Fix: Calendar: 4.1: Calendar Items with non-standard CLASS elements were processed incorrectly.
- Bug Fix: Mailbox: 4.3c4: flag changing operations in mdir-type Mailboxes could result in link failures.

**4.3.5** 23-Jun-05
`Valid Core License Keys:` issued on or after 1st of May'2003.

- CLUSTER: Remote Queue Submit settings are implemented (Settings->Cluster page).
- SMTP: the SubmitAddress/Domain envelope data is always recorded now.
- Bug Fix: HTTP: 4.3.3: Requests to Personal WebSite URLs with trailing "/" did not return the proper Content-

Types.

- Bug Fix: HTTP: 4.3c4: the Event Handlers could not be updated via the WebAdmin Interface.
- Bug Fix: Presence: 4.3.2: the non-standard "Idle" state was processed incorrectly.
- Bug Fix: SIP: 4.3c3: the Authenticate and Substitute 'From' External Gateway options were always on.
- Bug Fix: Cluster: 4.3c1: Remote Queue Submit operations did not check for all return codes.
- Bug Fix: WebUser: 4.1: Export of vCard/iCalendar data from submailboxes did not work correctly.
- Bug Fix: WebUser: 4.3: if a Domain did not have any S/MIME signing cert, the "Generate Key/Certificate" account operation produced a key without a Certificate.
- Bug Fix: WebUser: 4.3: updated alias records still could produce false "duplicate name" panics during startup.

### 4.3.4 30-May-05
`Valid Core License Keys:` issued on or after 1st of May'2003.

- Bug Fix: Domains: 4.3: when all account aliases were removed, the aliases.data file was not updated.

### 4.3.3 28-May-05
`Valid Core License Keys:` issued on or after 1st of May'2003.

- Platform: the FreeBSD5/x86_64 (AMD64, EMT64) version is released.
- Security: External AUTH Interface v.7 is implemented (the PLAIN response is supported now).
- RULES: the Current Account Address macro (^r) is implemented for the Reply and React With operations.
- RULES: multi-line header fields added with the Add Header operation are unwrapped before being used in the Header Field conditions.
- Admin: the KerberosKeys Domain Access right has been added.
- Migration: the --fixLongLines MoveIMAPMail utility option is implemented.
- WebUser: the initial default WebSkin name is now set to "Viewpoint".
- Bug Fix: CLUSTER: 4.3.1: Requests to Personal WebSite URLs with trailing "/" did not return the proper Content-Types.
- Bug Fix: Kernel: 4.3c2: Mailbox references to accounts with the same name in other domains did not work properly.
- Bug Fix: Kerberos: 4.3c1: Kerberos Keys for Directory-based Domains were not restored after restart.
- Bug Fix: LDAP: 4.3c1: non-plain (extended) record attributes were sent incorrectly.

### 4.3.2 20-May-05
`Valid Core License Keys:` issued on or after 1st of May'2003.

- Presence: Windows Messenger SERVICE requests are now used to obtain Presence info.
- RULES,PIPE: Rule-added header fields are now passed to external tasks if the [FILE] prefix is specified.
- PIPE,POP,RPOP: multi-line routing headers (Envelope-To, To/Cc) are supported now.
- Bug Fix: WebAdmin: 4.3c5: aliases specified in account import files caused import errors.
- Bug Fix: Domains: 4.3c3: account storage re-calculation could crash if the account info contained bogus data.
- Bug Fix: LIST: 4.3c4: the "X-List-Report" header field was composed incorrectly.

### 4.3.1 13-May-05
`Valid Core License Keys:` issued on or after 1st of May'2003.

- Rules: now the Redirect/Forward Rule Actions preserve the message Return-Path address if it is empty.
- SIP: the Presence Server now provides compatible "offline" status reports for the Microsoft (RTC) clients.
- CLI: the GetTempBlacklistedIPs command is implemented.
- Workaround: Windows: 4.3: a compiler bug caused garbage/crashes in the Rule "Store In" action logging.
- Bug Fix: Domains: 4.3c5: forwarder/alias list rebuilding (on startup/failover) could fail if names of recently deleted forwarders/aliases were used for new objects of a different type.
- Bug Fix: Domains: 4.3c5: forwarder/alias list rebuilding (on startup/failover) could fail if a file line started with a tabulation symbol.
- Bug Fix: Queue: 4.3: messages with the "new" SMTP Relay address did not release the Domain object.

- Bug Fix: TLS: 4.3c5: if a Domain did not have a PrivateKey, but it had its PKI services enabled, access to that Domain could crash the server.
- Bug Fix: WebUser: 3.5: Mailbox renaming operation could incorrectly rename Mailbox subscription elements.
- Bug Fix: CLUSTER: 4.0: Personal WebSite URLs were passed to backends with "URL escapes" removed.

**4.3** 03-May-05
`Valid Core License Keys:` issued on or after 1st of May'2003.

- SIP: RFC3903, RFC3863 are implemented.
- SIP: the Presence Server is implemented.
- SIGNAL: the Signal to Unknown Names Domain settings are implemented.
- Security: the Common Name menu for Domain Certificates now contains the "wildcard name" option.
- Queue: the SMTP Relay address for messages is stored as a Domain Name now. Cluster frontends now can use Domain IPs for messages generated with other cluster members.
- MAPI: the Encrypt operation is supported now.
- MAPI: the version 1.1.22 of the MAPI Connector is included.
- Bug Fix: WebUser: 4.3c5: the Decrypt action always stored messages with CR-LF EOLs causing problems on Unix and VMS platforms.

**4.3c5** 20-Apr-05
`Valid Core License Keys:` issued on or after 1st of April'2003.

- Domains: the Forwarder and Alias management for regular domains has been redesigned (10,000+ object sets can be updated quickly now).
- Platform: the Windows/Itanium version is released.
- Platform: the --SharedFiles startup parameter is enabled by default on Windows NT/XP/200x platforms. The --NoSharedFiles startup parameter is implemented.
- SMTP: the Always Use EHLO option is implemented.
- SysAdmin: now network ranges can be specified as `address/mask`.
- Security: Domain S/MIME settings are implemented.
- WebUser: the ShowLinks MessageViewer option is implemented.
- WebUser: vCard ADR field processing has been changed to provide better compatibility with Microsoft Outlook.
- WebUser: S/MIME certificates issued by the Server can be refreshed now.
- WebUser: the Encrypt and Decrypt operations are implemented now.
- RULES: the Security condition is implemented.
- RULES: the Store Encrypted action is implemented.
- RULES/WebUser: the Forward operations now use the From: field specified in the Account WebUser Preferences.
- HTTP: the HTTP Header Fields option for CGI processing is implemented.
- CLUSTER: frontend TLS pass-through decrypting improved to handle large data block sequences.
- Kernel: now MIME-encoded header fields can be generated using the binary (base64) encoding.
- SIP: now far-end NAT traversal does not use "sips" Record-Routes for TLS connections.
- Bug Fix: RADIUS: 4.3c4: MS-CHAP2 did not work for any addresses but the Main Domain account names.
- Bug Fix: RULES: 4.1: the Mark Redirected operation actually cleared the Redirected flag.

**4.3c4** 05-Apr-05
`Valid Core License Keys:` issued on or after 1st of April'2003.

- RULES: the Tag Subject operation is implemented.
- Kernel: message processing redesigned to support added headers and subject tags in all operations.
- Security: Heimdal Kerberos keytab files are accepted now.
- Security: External AUTH protocol version 6 is implemented.
- SMTP: the Hide Received fields option is implemented.

SMTP: RFC3865 ("No Soliciting") is implemented.
- SMTP: RFC3848 is implemented.
- SMTP: RFC2033 (the LMTP protocol) is implemented.
- MAPI: the version 1.1.20 of the MAPI Connector is included.
- RADIUS: MS-CHAPv1 and MS-CHAPv2 MPPE key generation is implemented.
- RADIUS: Digest authentication is implemented.
- Mailboxes: the Locked attribute is implemented. See the Objects->Mailboxes section for the details.
- Queue: now multiple address-level temp-errors generate only one Warning message.
- Router: the "Enabled" "detailed" address (name+detail) processing has changed to keep detailing on forwarded addresses.
- Foundation: the --DefaultStackSize startup parameter is implemented.
- Kernel: the --CreateTempFilesDirectly startup parameter is implemented.
- Calendar: now duration = 0 is accepted in iCALENDAR objects.
- WebUser: all internal routines now use the selected TimeZone data rather than the current TimeZone shift.
- WebAdmin: the Storage Used field on the Account Settings page is displayed as a button that can be used for re-synching the used storage counter.
- FTP: relative path names starting with ".." are supported (Unix-style).
- Bug Fix: TLS: 4.3c2: outgoing connections utilizing SSLv2 clientHello could crash the server.
- Bug Fix: HTTP: 4.3c1: "pure" NTLM authentication could fail.

**4.3c3** 15-Mar-05
`Valid Core License Keys`: issued on or after 1st of March'2003.

- Platform: SCO OpenServer/Intel version is released.
- SIGNAL: multiple AORs can be processed now. Full support for Redirect responses is implemented.
- SIP: Authorization for INVITEs sent to External Gateways is implemented.
- SIP: TLS Route URIs are supported now.
- CALENDAR: the Publish/Reply operations can now add the CN attribute to a new attendee.
- MAPI: the version 1.1.19 of the MAPI Connector is included.
- Foundation: the --NoThreadPriority startup parameter is implemented.
- CLI: the FindWebUserSession command is implemented.
- IMAP: Queue disk full situations could cause the APPEND command to add zero-size messages to Mailboxes.
- TLS: the Weak Ciphers option is implemented.
- LIST: the Service Fields Posting Policy option is implemented.
- LIST: the To And Cc option processing has been extended.
- LIST: the ^B (message sequence number) text macro is implemented.
- Directory: the STARTTLS option for Remote Units has been re-implemented according to RFC2830.
- Bug Fix: CALENDAR: RRULE parsing could stop on the UNTIL parameter.
- Bug Fix: HTTP: 4.2.3: inter-cluster Keep-Alive processing was broken.
- Bug Fix: HELPERS: 4.2: failed helpers were restarted even when the Use option was disabled.
- Bug Fix: LIST: 4.0: certain multipart messages sent to a list could crash the server.

**4.3c2** 18-Feb-05
`Valid Core License Keys`: issued on or after 1st of February'2003.

- SIP: the "far-end" NAT traversal functionality is implemented.
- SIP: MediaProxy mechanism redesigned to support call flows crossing the same edge server multiple times.
- SIP: RFC3892 (Referred-By) is implemented.
- SIP: support for session update re-INVITEs with multiple media streams added to NAT/Firewall proxies.
- SIP: Outgoing Proxy support is implemented.
- WebUser: the Skins developed by David Bakkers are included into the basic distribution.
- WebUser: the MakeCall feature has been implemented (calling a phone number listed in Contacts via your SIP devices).
- MAPI: the version 1.1.18 of the MAPI Connector is included. The MakeCall feature is implemented.

- WebAdmin: the Settings section rearranged: the Server-Wide Rules and RFC822 Receiver pages are moved to the Queue subsection. The Protection section renamed into the Network section.
- WebAdmin: the Server-wide and Cluster-wide Rule pages now provide the Comment field.
- WebAdmin: the NATed IPs page is added to Settings->Network subsection.
- Router: the .via and .relay domain name suffixes are implemented. The .smtp and .smtpq suffices are still supported, but deprecated.
- QUEUE: now the Send Return-Receipt To option controls the Relay reports, too.
- RADIUS: the MS-CHAPv1 and MS-CHAPv2 authentication methods are supported (enabled Windows VPN clients).
- Security: workarounds for Netscape bugs in its PLAIN authentication method are reimplemented.
- Security: the DIGEST authentication method for connection-oriented session re-implemented. It now accepts unquoted parameters (such as those sent with Java LDAP modules), and it can be used in Clusters (after all servers are upgraded).
- Foundation: the --LocalIPBuffer startup parameter is implemented.
- Bug Fix: LDAP: 4.0: response to a final DIGEST-MD5 SASL challenge was processed incorrectly.
- Bug Fix: SMTP: 4.1: initial prompt timeouts for "last" relays were shorter than timeouts for other relays.
- Bug Fix: GROUPS: 3.2: the Remove Author from Distribution option could not be disabled in Directory-based Domains.

**4.3c1** 07-Feb-05

`Valid Core License Keys:` issued on or after 1st of February'2003.

- ROUTER: Routing to the Forwarded To server now applies to mail-type routes only.
- SMTP: delay times for "temporarily rejected" messages and recipients are customizable now.
- SMTP: now the SPF records are not checked for Client and White Hole IP Addresses.
- Kernel: RFC2478 has been implemented.
- MAPI: the version 1.1.15 of the MAPI Connector is included.
- Security: the Advertise Clear Text methods option is implemented. It also implements the LOGINDISABLED capability in IMAP.
- Security: the Impersonate Login feature has been implemented (for PLAIN and GSSAPI login methods).
- Lawful Intercept: the Partial option is supported now.
- LDAP: authentication internals changed to better support LDAP provisioning in Cluster domains.
- WebUser: the Browser Authentication Login method is implemented (see the WebMail section for the details).
- HTTP: the CalendarDataDel realm is supported now (same as the CalendarData realm, but PUT operations delete all existing iCalendar items first).
- HTTP: the CONTENT_TYPE environment variable composed for CGIs now includes the boundary parameter.
- RULES: the Reply and React operations now use the From: and Charset settings specified in the Account WebUser Preferences.
- SIP: the server-generated Via header fields now contain the server SIP port numbers.
- CLUSTER: HTTP inter-server communication changed to support multi-step authentication schemes.
- TLS: now empty certificate sets and/or warning alerts are sent in response for "certificate requests" from remote servers.
- Bug Fix: WebUser: 4.1: header fields with ISO-2022 data could be composed incorrectly.

---

# Summary

**Platform**

- the FreeBSD5/x86_64 (AMD64, EMT64) version is released.
- the Windows/Itanium version is released.
- SCO OpenServer/Intel version is released.
- the --SharedFiles startup parameter is enabled by default on Windows NT/XP/200x platforms. The --

NoSharedFiles startup parameter is implemented.

## Foundation

- the --DefaultStackSize startup parameter is implemented.
- the --NoThreadPriority startup parameter is implemented.
- the --LocalIPBuffer startup parameter is implemented.

## Kernel

- now MIME-encoded header fields can be generated using the binary (base64) encoding.
- message processing redesigned to support added headers and subject tags in all operations.
- the --CreateTempFilesDirectly startup parameter is implemented.
- RFC2478 has been implemented.
- now network ranges can be specified as `address/mask`.

## ROUTER

- the "Enabled" "detailed" address (name+detail) processing has changed to keep detailing on forwarded addresses.
- the .via and .relay domain name suffixes are implemented. The .smtp and .smtpq suffices are still supported, but deprecated.
- routing to the Forwarded To server now applies to mail-type routes only.

## QUEUE

- the SMTP Relay address for messages is stored as a Domain Name now. Cluster frontends now can use Domain IPs for messages generated with other cluster members.
- now multiple address-level temp-errors generate only one Warning message.
- now the Send Return-Receipt To option controls the Relay reports, too.

## RULES

- Rule-added header fields are now passed to external tasks if the [FILE] prefix is specified.
- the Reply and React operations now use the From: and Charset settings specified in the Account WebUser Preferences.
- the Current Account Address macro (^r) is implemented for the Reply and React With operations.
- multi-line header fields added with the Add Header operation are unwrapped before being used in the Header Field conditions.
- now the Redirect/Forward Rule Actions preserve the message Return-Path address if it is empty.
- the Security condition is implemented.
- the Store Encrypted action is implemented.
- the Forward operations now use the From: field specified in the Account WebUser Preferences.
- the Tag Subject operation is implemented.

## Domains

- the Forwarder and Alias management for regular domains has been redesigned (10,000+ object sets can be updated quickly now).

## Mailboxes

- the Locked attribute is implemented. See the Objects->Mailboxes section for the details.

## Calendar

now duration = 0 is accepted in iCALENDAR objects.

- the Publish/Reply operations can now add the CN attribute to a new attendee.

## Security

- External AUTH Interface v.7 is implemented (the PLAIN response is supported now).
- the Common Name menu for Domain Certificates now contains the "wildcard name" option.
- Domain S/MIME settings are implemented.
- Heimdal Kerberos keytab files are accepted now.
- the DIGEST authentication method for connection-oriented session re-implemented. It now accepts unquoted parameters (such as those sent with Java LDAP modules), and it can be used in Clusters (after all servers are upgraded).
- the Advertise Clear Text methods option is implemented. It also implements the LOGINDISABLED capability in IMAP.
- the Impersonate Login feature has been implemented (for PLAIN and GSSAPI login methods).

## TLS

- now empty certificate sets and/or warning alerts are sent in response for "certificate requests" from remote servers.
- the Weak Ciphers option is implemented.

## Lawful Intercept

- the Partial option is supported now.

## WebAdmin

- the KerberosKeys Domain Access right has been added.
- the Storage Used field on the Account Settings page is displayed as a button that can be used for re-synching the used storage counter.
- the Settings section rearranged: the Server-Wide Rules and RFC822 Receiver pages are moved to the Queue subsection. The Protection section renamed into the Network section.
- the Server-wide and Cluster-wide Rule pages now provide the Comment field.
- the NATed IPs page is added to Settings->Network subsection.

## CLI

- the GetTempBlacklistedIPs command is implemented.
- the FindWebUserSession command is implemented.

## Directory

- the STARTTLS option for Remote Units has been re-implemented according to RFC2830.

## LDAP

- authentication internals changed to better support LDAP provisioning in Cluster domains.

## HTTP

- the HTTP Header Fields option for CGI processing is implemented.
- the CalendarDataDel realm is supported now (same as the CalendarData realm, but PUT operations delete all existing iCalendar items first).
- the CONTENT_TYPE environment variable composed for CGIs now includes the boundary parameter.

## FTP

- relative path names starting with ".." are supported (Unix-style).

## MAPI

- the version 1.1.22 of the MAPI Connector is included.
- the Encrypt operation is supported now.
- the MakeCall feature is implemented.

## WebUser

- the initial default WebSkin name is now set to "Viewpoint".
- the ShowLinks MessageViewer option is implemented.
- vCard ADR field processing has been changed to provide better compatibility with Microsoft Outlook.
- S/MIME certificates issued by the Server can be refreshed now.
- the Encrypt and Decrypt operations are implemented now.
- all internal routines now use the selected TimeZone data rather than the current TimeZone shift.
- the Skins developed by David Bakkers are included into the basic distribution.
- the MakeCall feature has been implemented (calling a phone number listed in Contacts via your SIP devices).
- the Browser Authentication Login method is implemented (see the WebMail section for the details).

## SIP

- RFC3903, RFC3863 are implemented.
- the Presence Server is implemented.
- Authorization for INVITEs sent to External Gateways is implemented.
- TLS Route URIs are supported now.
- the "far-end" NAT traversal functionality is implemented.
- MediaProxy mechanism redesigned to support call flows crossing the same edge server multiple times.
- RFC3892 (Referred-By) is implemented.
- support for session update re-INVITEs with multiple media streams added to NAT/Firewall proxies.
- Outgoing Proxy support is implemented.
- the server-generated Via header fields now contain the server SIP port numbers.

## SIGNAL

- the Signal to Unknown Names Domain settings are implemented.
- multiple AORs can be processed now. Full support for Redirect responses is implemented.

## CLUSTER

- frontend TLS pass-through decrypting improved to handle large data block sequences.
- HTTP inter-server communication changed to support multi-step authentication schemes.

## LIST

- the Service Fields Posting Policy option is implemented.
- the To And Cc option processing has been extended.
- the ^B (message sequence number) text macro is implemented.

## SMTP

- the Always Use EHLO option is implemented.
- the Hide Received fields option is implemented.

RFC3865 ("No Soliciting") is implemented.
- RFC3848 is implemented.
- RFC2033 (the LMTP protocol) is implemented.
- delay times for "temporarily rejected" messages and recipients are customizable now.
- now the SPF records are not checked for Client and White Hole IP Addresses.

## RADIUS

- MS-CHAPv1 and MS-CHAPv2 MPPE key generation is implemented.
- Digest authentication is implemented.
- the MS-CHAPv1 and MS-CHAPv2 authentication methods are supported (enabled Windows VPN clients).

## PIPE

- Rule-added header fields are now passed to external tasks if the [FILE] prefix is specified.
- multi-line routing headers (Envelope-To, To/Cc) are supported now.

## RPOP

- multi-line routing headers (Envelope-To, To/Cc) are supported now.

## Migration

- the --fixLongLines MoveIMAPMail utility option is implemented.

## Bug Fixes

- Bug Fix: CLUSTER: 4.0: Personal WebSite URLs were passed to backends with "URL escapes" removed.
- Bug Fix: CALENDAR: RRULE parsing could stop on the UNTIL parameter.
- Bug Fix: RULES: 4.1: the Mark Redirected operation actually cleared the Redirected flag.
- Bug Fix: HELPERS: 4.2: failed helpers were restarted even when the Use option was disabled.
- Bug Fix: LIST: 4.0: certain multipart messages sent to a list could crash the server.
- Bug Fix: LDAP: 4.0: response to a final DIGEST-MD5 SASL challenge was processed incorrectly.
- Bug Fix: SMTP: 4.1: initial prompt timeouts for "last" relays were shorter than timeouts for other relays.
- Bug Fix: GROUPS: 3.2: the Remove Author from Distribution option could not be disabled in Directory-based Domains.
- Bug Fix: WebUser: 3.5: Mailbox renaming operation could incorrectly rename Mailbox subscription elements.
- Bug Fix: WebUser: 4.1: header fields with ISO-2022 data could be composed incorrectly.
- Bug Fix: HTTP: 4.2.3: inter-cluster Keep-Alive processing was broken.

| Introduction | Features | **History** | How To | Help Me |
|---|---|---|---|---|

# Version 4.2 Revision History

- 6.2
- 6.1
- 6.0
- 5.4
- 5.3
- 5.2
- 5.1
- 5.0
- 4.3
- 4.2: Summary
- 4.1
- 4.0
- 3.x
- 2.x
- 1.x

RSS

**4.2.10** 15-Mar-05

`Valid Core License Keys:` issued on or after 15th of November'2001.

- Bug Fix: LDAP: 4.0: response to a final DIGEST-MD5 SASL challenge was processed incorrectly.
- Bug Fix: SMTP: 4.1: initial prompt timeouts for "last" relays were shorter than timeouts for other relays.
- Bug Fix: GROUPS: 3.2: the Remove Author from Distribution option could not be disabled in Directory-based Domains.
- Bug Fix: CALENDAR: RRULE parsing could stop on the UNTIL parameter.
- Bug Fix: HTTP: 4.2.3: inter-cluster Keep-Alive processing was broken.
- Bug Fix: HELPERS: 4.2: failed helpers were restarted even when the Use option was disabled.
- Bug Fix: LIST: 4.0: certain multipart messages sent to a list could crash the server.

**4.2.9** 04-Feb-05

- SMTP: The Original Return-Path domain is used for DNS verifications when the Forwarded To option is enabled.
- Bug Fix: WebUser: 4.2.7: under heavy load repeated failed access attempts to a closing session could crash the server.
- Bug Fix: SIP: 4.2: transport=tls URI parameters were processed incorrectly.

**4.2.8** 30-Dec-04

- SIP: External Gateway support has been added.
- SIP: Media Proxies now support re-INVITE requests.

SIP: Media Proxy Loops are detected and eliminated.
- FreeBusy: when Free/Busy info is built, the time zones associated with the Events are used instead of the default Server zone.
- Calendar: Recurrent event conflicts are detected now.
- SMTP: the Blocked Addresses to Remember option has been implemented.
- SMTP: now warning messages are sent when individual recipients are delayed.
- CLUSTER: the outgoing SMTP channels now use the Cluster name in the HELO/EHLO commands.
- LDAP: now LDAP Provisioning allows a client to retrieve effective settings directly from the Account Manager.
- Rules: recurrent Events can be auto-accepted now.
- WebUser: Recurrent event display method changed, the calendarpart.wssp file updated.
- WebUser: the Arabic dictionary file has been added. The RTL/LTR tags added to all Stock Skin pages to support Right-to-Left scripts and languages, the "ralign" elements added to the Mailboxes.wssp page.
- WebUser: the German and Dutch dictionary files have been added.
- WebUser: information about event conflicts is displayed (using the statusCode, conflictingID variables).
- MAPI: the version 1.1.12 of the MAPI Connector is included. Custom From and Reply-To addresses are supported again, with WebUser Interface integration.
- TFTP: ERROR packets are processed now.
- Kernel: all components (LIST, RULEs, etc.) now write their tags that can be checked with the Submit Address Rule condition. See the Transfer section for the details.
- Bug Fix: CALENDAR: the HTTP Delete operation (used to erase all calendar items) could crash the server.
- Bug Fix: RULES: 4.1: Rule Comments could be lost if Rule names/priorities were modified.
- Bug Fix: SIP: 4.2: parsing failed on address-specs with a comma sign in the comment field.
- Bug Fix: SMTP: 3.0: AUTH-ed messages were not relayed if the Relay to Non-Client IPs setting was set to Nobody.
- Bug Fix: SMTP: 4.2b8: malformed SPF records could cause infinite loops in macro processing.
- Bug Fix: WebUser 4.2.4: not all stateless request processors supported the <REDIRECT> tags in responses.

**4.2.7** 28-Nov-04

- Platforms: the OpenVMS/Alpha and OpenVMS/IA64 versions are released.
- WebUser: the Chinese (Simplified) and Slovak dictionary files have been added.
- Calendar: Data Import and HTTP Publishing operations now support CANCEL items. They can be used to remove existing items from the target Mailbox.
- Lawful Intercept: the Mailbox option is supported now.
- GROUP: messages distributed via Groups now get the X-Autogenerated: group header (so they are not considered Human-Generated).
- MAPI: the version 1.1.10 of the MAPI Connector is included. Kerberos Authentication is supported now.
- TFTP: excessive data bytes in ACK packets are ignored now.
- External Tasks: non-waiting "kills" are employed to avoid problems with some Linux kernels.
- WebUser: Directory Address Book now put RDNs into the attribute list to comply with LDAP standards.
- WebUser: Directory Address Book search function now checks the cn, mail, and uid attributes.
- SMTP: special processing of the port 587 (RFC2476) is implemented.
- CLUSTER: SMTP module now checks remote peers IP addresses against the IP addresses assigned to local domains (to avoid loops via load balancer IPs).
- Bug Fix: WebUser: 4.1: Directory Address Books with non-ASCII names did not work properly.
- Bug Fix: Queue: 4.1: If a message was submitted immediately after server startup (most likely - via the PIPE module) it could be left unprocessed till the next server restart.
- Bug Fix: Calendar: 4.1: when recurrent events were stored, the DTEND property was not stored in the local time format.

**4.2.6** 29-Oct-04

- KERBEROS: "principal" comparison rules were relaxed (to support Active Directory KDC).
- MAPI: the version 1.1.9 of the MAPI Connector is included. Kerberos Authentication is supported now.

- SIGNAL: requests for Live Communications can be be sent to group objects.
- GROUPS: the Disable Email and Disable Live Communication options are implemented.
- LIST: if the Hide From addresses option is selected, the Reply-To addresses are removed from messages.
- WebUser: the "style" tag parameters are removed when displaying HTML messages, as these parameters can contain scripts.
- WebUser: the Portuguese language support has been added.
- IMAP: the Content-Location data extension is added to the BODYSTRUCTURE output.
- Foundation: the getpwnam_r buffer increased to avoid crashes in some versions of Linux system libraries.
- Foundation: ISO-2022-JP encoding routines have been updated.
- Bug Fix: TFTP: 4.2.4-4.2.5: files containing an even number of 512 data blocks were transferred incorrectly.
- Bug Fix: CLUSTER: 4.2.3: creating the first Group/Forwarder in a freshly created Domain could crash the non-controller members.

**4.2.5** 10-Oct-04

- RULES: now the [bcc] prefix is supported in the Redirect/Forward actions.
- MAPI: the version 1.1.8 of the MAPI Connector is included. Multiple-language support is added to the client and server components. Now the localization file is always installed with the client component.
- Bug Fix: LIST: 4.2: "report to owner" messages had malformed From: header fields.
- Bug Fix: TFTP: 4.2.4: TFTP file reading was broken.
- Bug Fix: SIP: 4.2.1: received and rport values were not added to Via headers.
- Bug Fix: RULES: the Accept Event action incorrectly processed Non-Busy events (such as All-Day Events).

**4.2.4** 01-Oct-04

- WebUser: the special WML and I-Mode Skins are supported now. WAP/WML and I-Mode/cHTML Parameters are added to WebUser Preferences. Basic default WML and IMode Skins are created. The WML portions are removed from the Basic Unnamed Skin. See the WebApp section of the manual for the details.
- WebUser: the special <REDIRECT> tags are supported now. These tags can be used to return HTTP redirect responses. See the AppCodes section for more details.
- CLI: the CREATEWEBUSERSESSION command got new optional parameters.
- PWD: now APOP-type prompts are sent to Cluster members even if the APOP authentication is switched off.
- Bug Fix: External AUTH: 4.2b4: the positions of the key and password SASL command parameters were swapped..
- Bug Fix: WebUser: HTML encoding for Shift-JIS incorrectly processed half-width katakana.
- Bug Fix: LIST: 4.2: the From: address of service messages was formed incorrectly (quote marks around the address strings).
- Bug Fix: CLI: 4.1b: the CREATEWEBUSERSESSION command incorrectly processed the WML parameter.
- Bug Fix: TFTP: 4.2b7: data blocks resent because of duplicated ACKs were corrupted.
- Bug Fix: Utils: 2.0: the sendmail utility incorrectly checked error codes.
- Bug Fix: CLUSTER: 4.2: sometimes the STORE command parameters were processed incorrectly.
- Bug Fix: Foundation: 4.2: the new "date" data elements were parsed incorrectly.

**4.2.3** 22-Sep-04

- WebUser: several fields have been added to the Certificate/Key export format, to support Windows 2000 Key import functions.
- WebUser: the French dictionary file has been added.
- DNR: now the resolver checks that results of MX search operations are not listed in the Dummy IP Addresses list.
- CLUSTER: now Submit Address conditions work in Account-level Rules, too.
- Bug Fix: HTTP: authorization routines incorrectly checked user access rights.
- Bug Fix: WebUser: 4.2.2: Calendar menus were displayed incorrectly.

**4.2.2** 17-Sep-04

- WebUser: all text strings are moved from the Basic Skin into its strings.data file, Basic and GoldenFleece Skins modified.
- WebUser: multi-lingual Skins are supported now (the Language WebUser Preference), some language files are included into the Basic Skin.
- WebUser: stateless requests now use the Skin specified in the Domain Default Account WebUser Preferences (rather than the Unnamed Skin).
- WebUser: URLs with "anchors" (such as `http://aaa.dom/xxx#anchor`) are supported now.
- MAPI: the version 1.1.7 of the MAPI Connector is included.
- WebAdmin: The "Open Active Log showing last" setting is implemented (a new Log Manager setting).
- SMTP: the Relay to Any IP Address and Accept Wakeup settings can be set to "nobody" now.
- CLUSTER: SMTP: messages directed to non-cluster accounts are not submitted remotely, thus avoiding inter-cluster loops.
- CLUSTER: reloading of shared domains during controller failover is handled by several threads now.
- LOCAL: now Over-Quota notice messages are generated using 2 different Server Strings: one when an incoming message has been delivered, and the other one - when an incoming message is too big.
- IMAP: now the untagged UIDNEXT response is returned for the SELECT and EXAMINE commands.
- Bug Fix: CLUSTER FTP/CLI: 4.2b3: the PUTWEBFILE command used to create a directory could return the 599 error code.
- Bug Fix: EXTAUTH: 4.2: the ROUTE answer could crash the server.
- Bug Fix: Kernel: some internal routines processed Japanese and Korean charsets incorrectly.
- Bug Fix: DOMAINS: 4.2b6: renaming a domain could case loss of its aliases.
- Bug Fix: CLUSTER: 4.2b1: when Remote Queue processing was enabled, message temp files were not properly released.
- Bug Fix: RULES: the Accept Rule action did not decode quoted-printable and base64-encoded MIME parts.

**4.2.1** 05-Sep-04

- SIP: the SIP module has been completely redesigned. The SIGNAL module (proxy and server functions) is implemented.
- SIGNAL: the Presence module has been implemented (used to exchange online status information with "buddies").
- WebUser: the presence.wssp page/code module is implemented.
- MAPI: the version 1.1.6 of the MAPI Connector is included.
- Kerberos: Triple-DES methods are supported now.
- WSSP: now the JavaScript prefix encodes data into Unicode and outputs Unicode symbols using the \u prefix.
- WebSite: the "is directory" flag is now checked before attempting to read a file.
- Foundation: "gethostbyname" OS calls have been removed on most platforms (were used only to get the server computer own name).
- Directory: Remote Units close connections to remote servers if an error occurs.
- Directory: WebAdmin LDIF/LMOD import routines have been redesigned.
- Directory: WebAdmin Delete Record operation has been implemented.
- Calendar: incorrectly formatted VTIMEZONE elements created by Apple's iCal are accepted now.
- Local: delivery to "direct" Mailboxes has been optimized.
- Security: the External Authenticator Protocol Version 5 is implemented: the Router-type prefixes for the NEW command, the [NORELAY] tag for the ROUTED response.
- Directory: LDIF/LMOD import of base64-encoded data is implemented.
- SMTP: now explicitly specified White Hole addresses are not temp-blacklisted.
- SMTP: the header field added for blacklisted addresses can contain macro-symbols now.
- Bug Fix: SIP: 4.2b4: SRV DNS records could not be processed.
- Bug Fix: FreeBusy: 4.2: changes in the date-time scanning routines caused "illegal format" errors.
- Bug Fix: WebUser: 4.1b: the X-Priority header field values were generated incorrectly.
- Bug Fix: WebAdmin: 4.2b2: Subscriber management routine did not use the current Administrator preferences.

- Bug Fix: Foundation: 2.8: A "BigNumber library" negative number processing error could cause self-check exceptions.
- Bug Fix: Migration: 3.0: the MoveIMAPMail utility could crash on 64-bit platforms.

**4.2** 28-Jul-04

- SMTP: SPF processing has been improved.
- SIP: CANCEL transaction processing has been improved.
- LDAP: the status of the "criticality" control extension flag is changed to "optional".
- VCARD: now the date-time parser accepts separator symbols.
- LOCAL: the algorithm for "Over Quote" notice message generation has been changed.
- RULES: the Accept operation now understands meeting requests [mis]formatted with Apple's iCal application.
- Bug Fix: RPOP: 4.2b1: NTLM/MSN login (for msn.com) did not work with "short" challenges received from an MSN POP server.
- Bug Fix: PWD: 4.2b5: cluster "proxied" connections did not work.
- Bug Fix: HTTP: 3.5: "dirty" SNMP counters could be added to http* statistical elements if an SSL/TLS connection could not be established.
- Bug Fix: WSSP: 4.2b2: the JAVASCRIPT tag did not work correctly.
- Bug Fix: LDAP: 3.5: the STARTTLS extended request was not processed correctly.

**4.2b8** 13-Jul-04

- Platforms: Linux/Itanium version is re-released (using modern hardware and libraries).
- FTP: "data channel" close optimization (introduced in 4.2b6) removed as a workaround for MacOS Fetch client bug.
- FTP: the AUTH SSL and AUTH TLS-P commands are supported now.
- SMTP: SPF processing enhanced (macro processing, redirect, Received-SPF header field).
- Directory: RFC2253 (DN syntax rules) is implemented.
- Calendar: iCalendar EXDATE fields are supported.
- Bug Fix: WebUser: 4.2b2: when sending a letter with more than one attachment, the attachment names were corrupted.
- Bug Fix: WebUser: 4.1: the TakeAddress operation could cause "soft deadlock".
- Bug Fix: WebAdmin: 4.2b1: size-type parameters in Mailing List settings did not accept the "M" and "K" suffixes.
- Bug Fix: RADIUS: 4.2b4: the Accounting Log file was created incorrectly.

**4.2b7** 07-Jul-04

- Platforms: FreeBSD 5.x/Intel version is released.
- TFTP: the TFTP module has been implemented.
- HTTP: when closing a Keep-Alive connection on time-out, the TLS "quit" packet is not sent (a workaround for MS IExplorer SSL/TLS bug).
- HTTP: validity checks added to /redirect/ and /sredirect/ realm processing.
- Router: the Mail:,Access:, and Live: record prefixes are supported now.
- WebUser: named Stock Skins are supported now. The "Golden Fleece" Stock Skin is included into the distribution.
- WebUser: the Private Item Event/Task switch is implemented.
- WebUser: the Reply All operation removes duplicate and "self" addresses from To and Cc fields.
- WSSP: the EQUALSNOCASE operation is implemented.
- MAPI: submitted messages are marked as "authenticated" if the Return-Path is routed to the E-mail address of the current user.
- IMAP: multi-APPEND now uses only one temp file for all received messages.
- TLS: extended support for duplex operations.
- Domains: new group member addresses in Directory-based Domains are stored as "groupMember" attributes.

Domains: "empty" records for Directory-based Domains got their own objectClasses that include their RDN attributes.

- SNMP: the imapMAPIActive element is implemented.
- Bug Fix: WebUser: 4.2b2: the Password Recovery page did not work.
- Bug Fix: SIP: 4.2b2: attempts to connect to the TLS port could crash the server.

**4.2b6** 19-Jun-04

- Mailbox: the TextMailbox Manager implements Mailbox indexing.
- Calendar: now the recurrent events are stored together with their Time Zone information, and they are interpreted "within" that Time Zone.
- MAPI: now the Refresh button on the Settings->General page reloads the server-side MAPI DLL.
- Rules: Domain Rule processing has been changed: all Domain Rules with priorities >5 are executed before Account Rules, with priorities <= 5 are executed after Account Rules.
- WebAdmin: When the current Log file page is opened, the initial "From" time is set to the current time minus 2 minutes.
- WebAdmin: the SMTP settings page has been redesigned (divided into 3 pages).
- SMTP: the SPF Verification functionality has been added.
- Kerberos: added a workaround for Apple's Mail.app bug (excessive bytes at the end of Kerberos tokens).
- WebFiles: now it is possible to read and write files in chunks.
- CLI: the GETWEBFILE and PUTWEBFILE commands have new parameters now.
- FTP: GSSAPI MIC,ENC,CCC commands are supported now.
- FTP: GSSAPI-based security for control and data channels has been implemented.
- FTP: now files are uploaded and downloaded in chunks, thus allowing large (100MB+) file handling in both single-server and Cluster environments.
- Lawful Intercept: changes in the Settings now take effect on the currently opened accounts, too.
- DOMAINS: alias processing has been redesigned to support large number (100,000+) of aliases efficiently.
- LDAP, Directory: all DN size limits have been increased to 1024 bytes.
- Bug Fix: WebAdmin: 4.2b3: numbers of List Subscriber bounces were displayed incorrectly.
- Bug Fix: WebUser: 4.2b4: the notAltText flag was not set correctly.
- Bug Fix: Utils: 4.2b4: the -f flag could crash the sendmail utility.
- Bug Fix: HTTPUser: 4.1: the /CalendarData/ login realm was specified incorrectly.
- Bug Fix: Calendar: 4.1: concurrent modifications of the same Calendar object could cause "soft deadlock".
- Bug Fix: Kernel: 4.2b5: zero-timeout reading operation could interrupt SSL/TLS channels on cluster frontends.
- Bug Fix: FTP: 4.2b4: "unbinding" data sockets broke "active connections" through firewalls.
- Bug Fix: CLI: web site access security problem has been fixed.

**4.2b5** 04-Jun-04

- SIP: Packet Dispatcher modified to avoid deadlocks under heavy loads.
- SMTP: when the Prompt Delay setting is set to non-zero, the Server checks for any data received from the sender before the prompt is displayed. It drops the connection if it sees "pre-prompt data".
- WebUser: "flowed" text formatting has been improved (locates non-original lines and removes trailing spaces from them).
- Domains: object management has been changed to avoid problems with concurrent domain deletion and domain access operations.
- Platform: on MacOSX/Darwin the default PWD port is set to 8106 instead of 106 to avoid conflicts with Apple's own "Password Server".
- HTTP: the "Personal WebSite Prefix" became a Domain Setting now. The "~" prefix now always works (in addition to the specified Prefix).
- HTTP: the "Default WebPage" became an Account Setting now.
- HTTP: if a URL w/o a trailing slash refers to a Personal Web Site subdirectory, the "redirect" HTTP response is generated.
- Bug Fix: CLUSTER: 4.2b4: messages with empty return-paths could not be delivered to backends.

- Bug Fix: CLUSTER: 4.2b4: POP3 sessions to backend accounts could show INBOXes as empty.
- Bug Fix: External Filters: 4.2b4: the new External processor did not process the "INTF" responses.
- Bug Fix: FTP: 3.5: "resuming" FTP uploads could crash the server.

**4.2b4** 31-May-04

- Security: the Kerberos Authentication method (RFC1510) has been implemented. The Security->Kerberos page has been added to the Domain Settings, the Kerberos: Allow to Use option has been added to Account Settings, the Advertise GSSAPI option has been added to the Obscure settings.
- Security: the GSSAPI SASL method is implemented.
- RADIUS: External Helper support is implemented (to check request attributes and to produce response attributes).
- RADIUS: Accounting Log is implemented.
- SIP: minor changes and workarounds for broken SIP clients.
- External AUTH: the protocol version 4 is implemented (quoted strings for passwords and keys).
- Mailbox: the concept of "hidden" messages is implemented; "hidden" messages can be seen only by those who have the Admin access right for the Mailbox.
- IMAP: support for $Hidden flag is implemented.
- CLUSTER: pre-locking is implemented to avoid problems with accounts being accessed by non-controller backends immediately before or after they are being created.
- Admin: now the IP Address List settings can contain "exclude addresses" lines.
- WebAdmin: the Test Address button has been added to the Client IPs and Blacklisted IPs pages.
- MAPI: the version 1.1.3 of the MAPI Connector is included.
- Kernel: Japanese EUC-JP and Shift-JIS character sets are supported now.
- SMTP: the `@12.34.56.78:` queue prefixes are not added to host names included into Hold Mail lists (to support ATRN retrieval for those hosts).
- Bug Fix: CLUSTER: 4.2b1: the Local Delivery module incorrectly processed optional message tags.
- Bug Fix: LIST: 4.2b1: quoted-printable header/trailer encoding procedure did not work correctly.
- Bug Fix: WebUser: 4.2b3: strings.data dictionary did not get the case-insensitive flag.
- Bug Fix: Utilities: 4.2b3: the 'sendmail' program did not process multi-line header fields.
- Bug Fix: Kernel: 4.1: the TimeZone GMT->local conversion routine did not process daylight saving times.
- Bug Fix: FTP: 4.0: the server tried to bind outgoing connections to a fixed port, thus creating problems for multiple "active" file transfers.

**4.2b3** 06-May-04

- SIP: Windows Messenger support logic has been changed to support intermediate proxies.
- SMTP: the alternative Keep Trying setting for messages with empty Return-Paths is implemented.
- SMTP: the Delay Prompt option is implemented.
- QUEUE: file opening algorithm has been changed to allow parallel file opening/closing.
- DOMAINS: Domain Foldering is implemented (can be specified on the Domain Defaults page).
- CLI: the GETDOMAINLOCATION command is implemented.
- LOCAL, Account: the Account Quota options have been moved from the Local Module settings into Account Settings and Server Strings.
- LOCAL, Account: the Over Quota Notice message functionality has been implemented.
- LDAP: RFC2891 (sorting) is implemented.
- LDAP: RFC2696 (paging) is implemented.
- WebUser: now XML tags are removed from the displayed HTML code to stop the XML-based scripting tricks.
- WebUser: now the strings.data file is cached only when it is needed.
- WebUser: FreeBusy file processing has been changed (Outlook 2003 is supported now).
- Platform: the Windows installer adds the "tcpip" dependency into the "Service" record it creates.
- Utilities: several minor changes in the `sendmail` utility.
- CLUSTER: inter-cluster communications have changed to support larger file uploads.
- CLUSTER: initial Domain loading (non-controller): domain renaming/deletion is supported now.

- Foundation: additional error checks added to handle "disk full" errors.
- Bug Fix: WebAdmin: 3.5: custom Domain Admin Preferences.html pages were ignored.
- Bug Fix: WebAdmin: 4.1: the UTF-8 "BOM" symbols were removed on upload syntax check only, and were not removed from the stored files.
- Bug Fix: Skins: 3.5: rapid and concurrent update operation of the same strings.data file could crash the server.
- Bug Fix: CLI: 3.5: the ClearDomainSkinCache/ClearServerSkinCache/ClearClusterSkinCache commands could crash the server.
- Bug Fix: LIST: 4.2b2: the Subscribers WebAdmin page did not correctly display the number of bounces.
- Bug Fix: WebUser: 4.0: the Mailbox Rename function could crash the server.

**4.2b2** 09-Apr-04

- Account: the SIP access mode setting is implemented.
- SMTP: a 2-seconds delay is introduced before sending any error-type response to a non-client sender.
- LOCAL: delivery of an account queue is now finished after 2 minutes, causing queue rescheduling, to avoid delivery thread usurpation by an account with a huge queue.
- IMAP: processing of the STATUS command has been changed to allow for custom STATUS flags (used in MAPI).
- WebUser: the DeleteAll operation is implemented on the mailbox.wssp page (not included into the Stock Skin), and on the mailboxsettings.wssp page.
- WebUser: the "disabled" option for the ShowHTML setting is supported now.
- WebUser: Show Images setting is implemented. It controls how the objects (images, etc.) with external references are processed.
- WebUser: the Apply to Sub-Folders option is implemented (mailboxsettings.wssp).
- WebUser: the Priority field can be added to regular (non-event) composed messages.
- WebSite: support for non-ASCII file and folder names is implemented (websitebody.wssp modified).
- WebAdmin: the Charset admin preference is supported on the Monitors pages now.
- WSSP: the JAVASCRIPT prefix is implemented.
- RULES: the Existing Mailbox condition is implemented.
- ROUTER: now if object is a forwarder to user@domain, mailbox#object addresses are routed to mailbox#user@domain addresses.
- DOMAINS: now Directory-based domains process forwarders and aliases locally, even if they have been created on a different "host" server (in a distributed domain or a Static Cluster).
- SIP: TCP proxying for "msdata" streams is implemented, providing NAT/Firewall support for the "Share Application", "Remote Assistance", and other TCP-based features of Windows Messenger.
- RADIUS: EAP-MD5 (RFC1994, RFC2284, RFC2869) is implemented.
- POP, IMAP, ACAP: after several login errors a delay is introduced into all responses, not only into the 'failed authentication' ones.
- DEQUEUER: the Send Return-Receipts To setting is implemented.
- LIST: message subjects are now MIME-decoded before being used for confirmation and digest subject checks.
- LIST: now Subject Prefix strings can contain special symbol combinations (^N, etc.).
- LIST: a Domain Administrator can modify List settings and subscriptions only if the CanAccessLists right is granted.
- LIST: the WebAdmin interface is redesigned to support Cluster-safe calls and the CanAccessList Domain Access Right.
- CLI: the PROCESSBOUNCE command is implemented.
- IMAP: the REPLACEACL command is implemented.
- SMTP: when a Return-Path address routes to a local Group or Mailing List, the Force AUTH option is applied, too.
- PIPE: submitted messages now get the "client/trusted" flag (instead of the "authenticated" flag they got before).
- Directory: Remote Units: the maximum acceptable DN size has been increased from 256 to 1024 bytes.
- Bug Fix: WebUser: 4.2b1: the Calendar realm could not be used because of mis-formed AUTH headers.
- Bug Fix: WebUser: 4.2b1: Calendar items were not recognized as "own" items.
- Bug Fix: WebSite: 4.2b1: account creation failed if the Account Template contained an Initial Home Web Page.

- Bug Fix: SIP: 4.2b1: parser incorrectly processed old-style "addr-specs" lexems.
- Bug Fix: IMAP: 2.0: the LISTRIGHTS response did not include the "required rights" element.

**4.2b1** 23-Mar-04

- SIP: SIP server/proxy is implemented (including NAT support and media proxy).
- SIP: support for non-standard protocol used in Windows Messenger 5.0 is implemented.
- Platform: the NetBSD/Intel version is released (NetBSD 2.0 is required).
- Protection: The LAN IPs settings have been implemented. The "Process LAN IP Addresses as Clients" setting has been implemented.
- CLI: The GETLANIPS,SETLANIPS,GETCLUSTERLANIPS,SETCLUSTERLANIPS commands have been implemented.
- Directory: the RFC2798 implemented (all additional inetOrgPerson attributes have been added to the built-in Local Unit schema).
- Queue: the queue directory "foldering" is implemented (see the Transfer section of the manual for the details).
- Queue: the limit on number of queued messages is implemented.
- WebAdmin: now the "date created" information is displayed on the Account Settings page.
- WebAdmin: several account aliases can be specified in one field (separated with comma signs).
- MAPI: MAPI connector 1.0.84 is included.
- WebUser: the References header field is supported now.
- WebUser: the TimeZone setting is implemented.
- WebUser: the QP and Base64-encoded calendar MIME parts are displayed now.
- WebUser: Directory-type Address Books now support the "Take Certificate" action.
- WebUser: Apple vCards (Unicode-formatted) are supported now.
- WebUser: now S/MIME certificates can be generated by users themselves.
- WebUser: the Password Recovery message does not contain the name of the account any more.
- CALENDAR: Task Reply messages now contain copies of the Task Description field.
- CLUSTER: Domains loading during a non-controller member startup has been improved.
- CLUSTER: Remote Queue processing has been implemented.
- Directory-Based Domains: the Template updating mechanism has been improved.
- Kernel: the ISO-8859-13,ISO-8859-14,ISO-8859-16 charsets are supported now.
- Kernel: the Windows extensions are added to the ISO-8859-x charsets.
- Kernel: charset processing has been modified: charset aliases are supported now.
- SNMP: the max size of generated responses is increased to 1400 bytes.
- IMAP: the SETQUOTA command is fully implemented.
- LIST: the feed trailer-footers are QP-encoded when they are inserted into a quoted-printable-encoded text.
- Rules: now the Mark action can be used in the Server-Wide and Cluster-Wide Rules.
- Foundation: the "object checking routines" do not slow down the system any more, the --NoObjReleaseCheck flag has been phased out.
- SMTP: Sender Disconnect after errors option now works for non-Client senders only.
- SMTP: the Recipients per time period Limit is implemented.
- SMTP: sending to a domain is rejected immediately if all domain MX records cannot be resolved because of non-timeout problems.
- Security: the DIGEST and NTLM authentication methods added to the HTTP modules.
- Security: the Advertise APOP method option is implemented (see the Security section of the manual).
- HTTP: if a CGI output starts with "HTTP/", no HTTP header data is added by the Server.
- HTTP: if a CGI output starts with the "Status:" field, that field it used to form the response code and response text.
- Bug Fix: WebUser: 4.1b: message with empty body and no attachment were sent w/o the Content-Type field and the header separator line.
- Bug Fix: WebUser: the Notes Mailboxes were given the IPF.stickyNote class instead of IPF.StickyNote, causing problem for MS Outlook.
- Bug Fix: CLI: a malformed WebFile command could crash the server.
- Bug Fix: CLI: the GETTEMPCLIENTIPS command could return garbage after the data.

- Bug Fix: WebUser: non-ASCII attachment names were encoded incorrectly if the selected charset was not UTF-8.
- Bug Fix: CLUSTER: the controller incorrectly checked if an account was in use on other backends when a rename/remove operation was requested.
- Bug Fix: Directory-Based Domains: if an account was auto-created (by means of adding a directory record) and the domain Template contained an Initial Message, the server could crash.

---

# Summary

### Platform

- the NetBSD/Intel version is released.
- the OpenVMS/Alpha and OpenVMS/IA64 versions are released.
- Linux/Itanium version is re-released (using modern hardware and libraries).
- FreeBSD 5.x/Intel version is released.
- on MacOSX/Darwin the default PWD port is set to 8106 instead of 106 to avoid conflicts with Apple's own "Password Server".
- the Windows installer adds the "tcpip" dependency into the "Service" record it creates.
- External Tasks: non-waiting "kills" are employed to avoid problems with some Linux kernels.

### Foundation

- the getpwnam_r buffer increased to avoid crashes in some versions of Linux system libraries.
- ISO-2022-JP encoding routines have been updated.
- "gethostbyname" OS calls have been removed on most platforms (were used only to get the server computer own name).
- additional error checks added to handle "disk full" errors.
- the "object checking routines" do not slow down the system any more, the --NoObjReleaseCheck flag has been phased out.

### Kernel

- all components (LIST, RULEs, etc.) now write their tags that can be checked with the Submit Address Rule condition. See the Transfer section for the details.
- Japanese EUC-JP and Shift-JIS character sets are supported now.
- the ISO-8859-13,ISO-8859-14,ISO-8859-16 charsets are supported now.
- the Windows extensions are added to the ISO-8859-x charsets.
- charset processing has been modified: charset aliases are supported now.
- now the IP Address List settings can contain "exclude addresses" lines.
- now the date-time parser accepts separator symbols.

### ROUTER

- the Mail:,Access:, and Live: record prefixes are supported now.
- now if object is a forwarder to user@domain, mailbox#object addresses are routed to mailbox#user@domain addresses.

### QUEUE

- file opening algorithm has been changed to allow parallel file opening/closing.
- the Queue directory "foldering" is implemented (see the Transfer section of the manual for the details).
- the limit on number of queued messages is implemented.
- the Send Return-Receipts To setting is implemented.

## Rules

- recurrent Events can be auto-accepted now.
- now the [bcc] prefix is supported in the Redirect/Forward actions.
- the Accept operation now understands meeting requests [mis]formatted with Apple's iCal application.
- Domain Rule processing has been changed: all Domain Rules with priorities >5 are executed before Account Rules, with priorities <= 5 are executed after Account Rules.
- the Existing Mailbox condition is implemented.
- now the Mark action can be used in the Server-Wide and Cluster-Wide Rules.

## Domains

- new group member addresses in Directory-based Domains are stored as "groupMember" attributes.
- "empty" records for Directory-based Domains got their own objectClasses that include their RDN attributes.
- alias processing has been redesigned to support large number (100,000+) of aliases efficiently.
- object management has been changed to avoid problems with concurrent domain deletion and domain access operations.
- Domain Foldering is implemented (can be specified on the Domain Defaults page).
- now Directory-based domains process forwarders and aliases locally, even if they have been created on a different "host" server (in a distributed domain or a Static Cluster).

## Accounts

- the SIP access mode setting is implemented.
- the Account Quota options have been moved from the Local Module settings into Account Settings and Server Strings.
- the Over Quota Notice message functionality has been implemented.

## Groups

- messages distributed via Groups now get the X-Autogenerated: group header (so they are not considered Human-Generated).
- the Disable Email and Disable Live Communication options are implemented.

## Mailboxes

- the concept of "hidden" messages is implemented; "hidden" messages can be seen only by those who have the Admin access right for the Mailbox.
- the TextMailbox Manager implements Mailbox indexing.

## WebSite

- now it is possible to read and write files in chunks.
- support for non-ASCII file and folder names is implemented (websitebody.wssp modified).
- the "is directory" flag is now checked before attempting to read a file.

## Calendar

- Task Reply messages now contain copies of the Task Description field.
- Recurrent event conflicts are detected now.
- Data Import and HTTP Publishing operations now support CANCEL items. They can be used to remove existing items from the target Mailbox.
- incorrectly formatted VTIMEZONE elements created by Apple's iCal are accepted now.
- iCalendar EXDATE fields are supported.
- now the recurrent events are stored together with their Time Zone information, and they are interpreted "within"

that Time Zone.
- when Free/Busy info is built, the time zones associated with the Events are used instead of the default Server zone.

## Protection

- POP, IMAP, ACAP: after several login errors a delay is introduced into all responses, not only into the 'failed authentication' ones.
- The LAN IPs settings have been implemented. The "Process LAN IP Addresses as Clients" setting has been implemented.

## Security

- the External Authenticator Protocol Version 5 is implemented: the Router-type prefixes for the NEW command, the [NORELAY] tag for the ROUTED response.
- the Kerberos Authentication method (RFC1510) has been implemented. The Security->Kerberos page has been added to the Domain Settings, the Kerberos: Allow to Use option has been added to Account Settings, the Advertise GSSAPI option has been added to the Obscure settings.
- the GSSAPI SASL method is implemented.
- the DIGEST and NTLM authentication methods added to the HTTP modules.
- the Advertise APOP method option is implemented (see the Security section of the manual).

## Kerberos

- "principal" comparison rules were relaxed (to support Active Directory KDC).
- Triple-DES methods are supported now.
- added a workaround for Apple's Mail.app bug (excessive bytes at the end of Kerberos tokens).

## TLS

- extended support for duplex operations.

## Lawful Intercept

- the Mailbox option is supported now.
- changes in the Settings now take effect on the currently opened accounts, too.

## WebAdmin

- the "Open Active Log showing last" setting is implemented (a new Log Manager setting).
- when the current Log file page is opened, the initial "From" time is set to the current time minus 2 minutes.
- the SMTP settings page has been redesigned (divided into 3 pages).
- the Test Address button has been added to the Client IPs and Blacklisted IPs pages.
- the Charset admin preference is supported on the Monitors pages now.
- now the "date created" information is displayed on the Account Settings page.
- several account aliases can be specified in one field (separated with comma signs).
- Directory-Based Domains: the Template updating mechanism has been improved.

## CLI

- the CREATEWEBUSERSESSION command got new optional parameters.
- the GETWEBFILE and PUTWEBFILE commands have new parameters now.
- the GETDOMAINLOCATION command is implemented.
- The GETLANIPS,SETLANIPS,GETCLUSTERLANIPS,SETCLUSTERLANIPS commands have been implemented.

- the PROCESSBOUNCE command is implemented.

**Directory**

- Remote Units close connections to remote servers if an error occurs.
- WebAdmin LDIF/LMOD import routines have been redesigned.
- WebAdmin Delete Record operation has been implemented.
- LDIF/LMOD import of base64-encoded data is implemented.
- RFC2253 (DN syntax rules) is implemented.
- all DN size limits have been increased to 1024 bytes.
- Remote Units: the maximum acceptable DN size has been increased from 256 to 1024 bytes.
- the RFC2798 implemented (all additional inetOrgPerson attributes have been added to the built-in Local Unit schema).

**LDAP**

- now LDAP Provisioning allows a client to retrieve effective settings directly from the Account Manager.
- the status of the "criticality" control extension flag is changed to "optional".
- RFC2891 (sorting) is implemented.
- RFC2696 (paging) is implemented.
- all DN size limits have been increased to 1024 bytes.

**HTTP**

- when closing a Keep-Alive connection on time-out, the TLS "quit" packet is not sent (a workaround for MS IExplorer SSL/TLS bug).
- validity checks added to /redirect/ and /sredirect/ realm processing.
- the "Personal WebSite Prefix" became a Domain Setting now. The "~" prefix now always works (in addition to the specified Prefix).
- the "Default WebPage" became an Account Setting now.
- if a URL w/o a trailing slash refers to a Personal File Site subdirectory, the "redirect" HTTP response is generated.
- if a CGI output starts with "HTTP/", no HTTP header data is added by the Server.
- if a CGI output starts with the "Status:" field, that field it used to form the response code and response text.

**FTP**

- "data channel" close optimization (introduced in 4.2b6) removed as a workaround for MacOS Fetch client bug.
- the AUTH SSL and AUTH TLS-P commands are supported now.
- GSSAPI MIC,ENC,CCC commands are supported now.
- GSSAPI-based security for control and data channels has been implemented.
- now files are uploaded and downloaded in chunks, thus allowing large (100MB+) file handling in both single-server and Cluster environments.

**TFTP**

- ERROR packets are processed now.
- excessive data bytes in ACK packets are ignored now.
- the TFTP module has been implemented.

**MAPI**

- the version 1.1.12 of the MAPI Connector is included.
- WebUser Interface settings integration is implemented.
- Kerberos Authentication is supported now.

- Multiple-language support is added to the client and server components. Now the localization file is always installed with the client component.
- submitted messages are marked as "authenticated" if the Return-Path is routed to the E-mail address of the current user.
- now the Refresh button on the Settings->General page reloads the server-side MAPI DLL.

**WebUser**

- Recurrent event display method changed, the calendarpart.wssp file updated.
- the Arabic dictionary file has been added. The RTL/LTR tags added to all Stock Skin pages to support Right-to-Left scripts and languages, the "ralign" elements added to the Mailboxes.wssp page.
- the German and Dutch dictionary files have been added.
- information about event conflicts is displayed (using the statusCode, conflictingID variables).
- the Chinese (Simplified) and Slovak dictionary files have been added.
- Directory Address Book now put RDNs into the attribute list to comply with LDAP standards.
- Directory Address Book search function now checks the cn, mail, and uid attributes.
- the "style" tag parameters are removed when displaying HTML messages, as these parameters can contain scripts.
- the Portuguese language support has been added.
- the special WML and I-Mode Skins are supported now. WAP/WML and I-Mode/cHTML Parameters are added to WebUser Preferences. Basic default WML and IMode Skins are created. The WML portions are removed from the Basic Unnamed Skin. See the WebApp section of the manual for the details.
- the special <REDIRECT> tags are supported now. These tags can be used to return HTTP redirect responses. See the AppCodes section for more details.
- several fields have been added to the Certificate/Key export format, to support Windows 2000 Key import functions.
- the French dictionary file has been added.
- all text strings are moved from the Basic Skin into its strings.data file, Basic and GoldenFleece Skins modified.
- multi-lingual Skins are supported now (the Language WebUser Preference), some language files are included into the Basic Skin.
- stateless requests now use the Skin specified in the Domain Default Account WebUser Preferences (rather than the Unnamed Skin).
- URLs with "anchors" (such as http://aaa.dom/xxx#anchor) are supported now.
- named Stock Skins are supported now. The "Golden Fleece" Stock Skin is included into the distribution.
- the Private Item Event/Task switch is implemented.
- the Reply All operation removes duplicate and "self" addresses from To and Cc fields.
- "flowed" text formatting has been improved (locates non-original lines and removes trailing spaces from them).
- now XML tags are removed from the displayed HTML code to stop the XML-based scripting tricks.
- now the strings.data file is cached only when it is needed.
- FreeBusy file processing has been changed (Outlook 2003 is supported now).
- the DeleteAll operation is implemented on the mailbox.wssp page (not included into the Stock Skin), and on the mailboxsettings.wssp page.
- the "disabled" option for the ShowHTML setting is supported now.
- Show Images setting is implemented. It controls how the objects (images, etc.) with external references are processed.
- the Apply to Sub-Folders option is implemented (mailboxsettings.wssp).
- the Priority field can be added to regular (non-event) composed messages.
- the References header field is supported now.
- the TimeZone setting is implemented.
- the QP and Base64-encoded calendar MIME parts are displayed now.
- Directory-type Address Books now support the "Take Certificate" action.
- Apple vCards (Unicode-formatted) are supported now.
- now S/MIME certificates can be generated by users themselves.
- the Password Recovery message does not contain the name of the account any more.

## WSSP

- the EQUALSNOCASE operation is implemented.
- the JAVASCRIPT prefix is implemented.

## SIP

- External Gateway support has been added.
- Media Proxies now support re-INVITE requests.
- Media Proxy Loops are detected and eliminated.
- TCP proxying for "msdata" streams is implemented, providing NAT/Firewall support for the "Share Application", "Remote Assistance", and other TCP-based features of Windows Messenger.
- SIP server/proxy is implemented (including NAT support and media proxy).
- support for non-standard protocol used in Windows Messenger 5.0 is implemented.

## CLUSTER

- the outgoing SMTP channels now use the Cluster name in the HELO/EHLO commands.
- SMTP module now checks remote peers IP addresses against the IP addresses assigned to local domains (to avoid loops via load balancer IPs).
- now Submit Address conditions work in Account-level Rules, too.
- SMTP: messages directed to non-cluster accounts are not submitted remotely, thus avoiding inter-cluster loops.
- reloading of shared domains during controller failover is handled by several threads now.
- pre-locking is implemented to avoid problems with accounts being accessed by non-controller backends immediately before or after they are being created.
- inter-cluster communications have changed to support larger file uploads.
- initial Domain loading (non-controller): domain renaming/deletion is supported now.
- Domains loading during a non-controller member startup has been improved.
- Remote Queue processing has been implemented.

## SNMP

- the imapMAPIActive element is implemented.

## LIST

- if the Hide From addresses option is selected, the Reply-To addresses are removed from messages.
- message subjects are now MIME-decoded before being used for confirmation and digest subject checks.
- now Subject Prefix strings can contain special symbol combinations (^N, etc.).
- a Domain Administrator can modify List settings and subscriptions only if the CanAccessLists right is granted.
- the WebAdmin interface is redesigned to support Cluster-safe calls and the CanAccessList Domain Access Right.
- the feed trailer-footers are QP-encoded when they are inserted into a quoted-printable-encoded text.

## IMAP

- the Content-Location data extension is added to the BODYSTRUCTURE output.
- now the untagged UIDNEXT response is returned for the SELECT and EXAMINE commands.
- multi-APPEND now uses only one temp file for all received messages.
- support for $Hidden flag is implemented.
- processing of the STATUS command has been changed to allow for custom STATUS flags (used in MAPI).
- the REPLACEACL command is implemented.
- the SETQUOTA command is fully implemented.

## SMTP

- The Original Return-Path domain is used for DNS verifications when the Forwarded To option is enabled.
- the Blocked Addresses to Remember option has been implemented.
- now warning messages are sent when individual recipients are delayed.
- special processing of the port 587 (RFC2476) is implemented.
- the Relay to Any IP Address and Accept Wakeup settings can be set to "nobody" now.
- now explicitly specified White Hole addresses are not temp-blacklisted.
- the header field added for blacklisted addresses can contain macro-symbols now.
- the SPF Verification functionality has been added.
- the `@12.34.56.78:` queue prefixes are not added to host names included into Hold Mail lists (to support ATRN retrieval for those hosts).
- the alternative Keep Trying setting for messages with empty Return-Paths is implemented.
- the Delay Prompt option is implemented.
- a 2-seconds delay is introduced before sending any error-type response to a non-client sender.
- when a Return-Path address routes to a local Group or Mailing List, the Force AUTH option is applied, too.
- Sender Disconnect after errors option now works for non-Client senders only.
- the Recipients per time period Limit is implemented.
- sending to a domain is rejected immediately if all domain MX records cannot be resolved because of non-timeout problems.

## LOCAL

- now Over-Quota notice messages are generated using 2 different Server Strings: one when an incoming message has been delivered, and the other one - when an incoming message is too big.
- delivery to "direct" Mailboxes has been optimized.
- the algorithm for "Over Quote" notice message generation has been changed.
- delivery of an account queue is now finished after 2 minutes, causing queue rescheduling, to avoid delivery thread usurpation by an account with a huge queue.

## RADIUS

- External Helper support is implemented (to check request attributes and to produce response attributes).
- Accounting Log is implemented.
- EAP-MD5 (RFC1994, RFC2284, RFC2869) is implemented.

## PIPE

- PIPE: submitted messages now get the "client/trusted" flag (instead of the "authenticated" flag they got before).

## PWD

- PWD: now APOP-type prompts are sent to Cluster members even if the APOP authentication is switched off.

## DNR

- DNR: now the resolver checks that results of MX search operations are not listed in the Dummy IP Addresses list.

## Utilities

- Utilities: several minor changes in the `sendmail` utility.

## Bug Fixes

- Bug Fix: Foundation: 2.8: A "BigNumber library" negative number processing error could cause self-check exceptions.
- Bug Fix: Kernel: 3.5: some internal routines processed Japanese and Korean charsets incorrectly.
- Bug Fix: Kernel: 4.1: the TimeZone GMT->local conversion routine did not process daylight saving times.
- Bug Fix: Calendar: 4.1: concurrent modifications of the same Calendar object could cause "soft deadlock".
- Bug Fix: Calendar: 4.1: when recurrent events were stored, the DTEND property was not stored in the local time format.
- Bug Fix: Calendar: RRULE parsing could stop on the UNTIL parameter.
- Bug Fix: Calendar: the HTTP Delete operation (used to erase all calendar items) could crash the server.
- Bug Fix: CLUSTER: the controller incorrectly checked if an account was in use on other backends when a rename/remove operation was requested.
- Bug Fix: GROUPS: 3.2: the Remove Author from Distribution option could not be disabled in Directory-based Domains.
- Bug Fix: Queue: 4.1: If a message was submitted immediately after server startup (most likely - via the PIPE module) it could be left unprocessed till the next server restart.
- Bug Fix: RULES: 4.1: Rule Comments could be lost if Rule names/priorities were modified.
- Bug Fix: RULES: 4.1: the Accept Event action incorrectly processed Non-Busy events (such as All-Day Events).
- Bug Fix: RULES: 4.1: the Accept Event action did not decode quoted-printable and base64-encoded MIME parts.
- Bug Fix: WebAdmin: 3.5: custom Domain Admin Preferences.html pages were ignored.
- Bug Fix: WebAdmin: 4.1: the UTF-8 "BOM" symbols were removed on upload syntax check only, and were not removed from the stored files.
- Bug Fix: CLI: 4.1b: the CREATEWEBUSERSESSION command incorrectly processed the WML parameter.
- Bug Fix: CLI: 3.5: a malformed WebFile command could crash the server.
- Bug Fix: CLI: 4.1: the GETTEMPCLIENTIPS command could return garbage after the data.
- Bug Fix: CLI: 4.1: web site access security problem has been fixed.
- Bug Fix: CLI: 3.5: the ClearDomainSkinCache/ClearServerSkinCache/ClearClusterSkinCache commands could crash the server.
- Bug Fix: LDAP: 3.5: the STARTTLS extended request was not processed correctly.
- Bug Fix: LDAP: 4.0: response to a final DIGEST-MD5 SASL challenge was processed incorrectly.
- Bug Fix: Skins: 3.5: rapid and concurrent update operation of the same strings.data file could crash the server.
- Bug Fix: WebUser: 4.1: HTML encoding for Shift-JIS incorrectly processed half-width katakana.
- Bug Fix: WebUser: 4.1b: the X-Priority header field values were generated incorrectly.
- Bug Fix: WebUser: 4.1: the TakeAddress operation could cause "soft deadlock".
- Bug Fix: WebUser: 4.0: the Mailbox Rename function could crash the server.
- Bug Fix: WebUser: 4.1: Directory Address Books with non-ASCII names did not work properly.
- Bug Fix: WebUser: 4.1b: message with empty body and no attachment were sent w/o the Content-Type field and the header separator line.
- Bug Fix: WebUser: 4.1: the Notes Mailboxes were given the `IPF.stickyNote` class instead of `IPF.StickyNote`, causing problem for MS Outlook.
- Bug Fix: WebUser: non-ASCII attachment names were encoded incorrectly if the selected charset was not UTF-8.
- Bug Fix: HTTP: 3.5: "dirty" SNMP counters could be added to http* statistical elements if an SSL/TLS connection could not be established.
- Bug Fix: HTTP: 4.1: the /CalendarData/ login realm was specified incorrectly.
- Bug Fix: FTP: 3.5: "resuming" FTP uploads could crash the server.
- Bug Fix: FTP: 4.0: the server tried to bind outgoing connections to a fixed port, thus creating problems for multiple "active" file transfers.
- Bug Fix: SMTP: 4.1: initial prompt timeouts for "last" relays were shorter than timeouts for other relays.
- Bug Fix: SMTP: 3.0: AUTH-ed messages were not relayed if the Relay to Non-Client IPs setting was set to Nobody.
- Bug Fix: LIST: 4.0: certain multipart messages sent to a list could crash the server.
- Bug Fix: IMAP: 2.0: the LISTRIGHTS response did not include the "required rights" element.

- Bug Fix: Migration: 3.0: the MoveIMAPMail utility could crash on 64-bit platforms.
- Bug Fix: Utils: 2.0: the sendmail utility incorrectly checked error codes.
- Bug Fix: Directory-Based Domains: 4.0: if an account was auto-created (by means of adding a directory record) and the domain Template contained an Initial Message, the server could crash.

CommuniGate **Pro**

| Introduction | Features | History | How To | Help Me |
|---|---|---|---|---|

# Version 4.1 Revision History

- 6.2
- 6.1
- 6.0
- 5.4
- 5.3
- 5.2
- 5.1
- 5.0
- 4.3
- 4.2
- 4.1: Summary
- 4.0
- 3.x
- 2.x
- 1.x

RSS

**4.1.8** 17-Nov-03

`Valid Core License Keys:` all.

- CLI: the WML parameter is added to the CreateWebUserSession command.
- WebUser: the multi-card vCard MIME parts are shown as attachments now, so they can be downloaded and imported.
- Bug Fix: WebUser: 4.1.7: Attempts to send a Read Receipt could lock the session and the Mailbox.
- Bug Fix: Cluster: 4.1.4: the CreateGroup operation worked, but returned the "unimplemented in this domain" error.
- Bug Fix: WebUser: 4.1.7: Attempts to add elements from Directory-based Address book could crash the server.

**4.1.7** 15-Nov-03

- Migration: the MS Exchange migration utility is released.
- MAPI: the documentation is updated.
- IMAP: the RFC3503 ($MDNSent flag) is implemented.
- WebUser: support for the $MDNSent message flag is implemented.
- WebUser: the Recipients Limit setting (for composed messages) is implemented.
- Helpers: the time-out and autorestart functions have been added to the External AUTHenticator.
- PIPE: the Envelope-Notify header field is supported now.
- Personal Site: upload file limit increased to 64MB, an error is generated when the limit is exceeded.
- CLUSTER: FTP operations in a Static Cluster are supported now.
- Bug Fix: ACAP: 4.1b: dropping a connection in the middle of certain operations could crash the server.
- Bug Fix: CLUSTER: 4.1: extremely long Alerts sent via POP could crash frontend servers.
- Bug Fix: Kernel: 1.0: URL encoding for the space symbol has been fixed.

- Bug Fix: WebUser: 4.1.1: Directory Address Books did show records with multi-value "mail" attributes, but could not retrieve those attributes properly.
- Bug Fix: Rules: 4.1.5: when a dataset had more than 500 elements so the Rule actions did not add new elements to it, the dataset was not released.
- Bug Fix: Platform: 4.0: OS/2 "no inheritance" bit was set incorrectly, causing "non-closing" stdin channels in child programs.

**4.1.6** 24-Oct-03

- Kernel: now the default Client IP Addresses list includes the networks the server belongs to.
- SMTP: the Relay to Non-Clients setting is now set to "If received from Clients only" by default.
- Calendar: VTIMEZONE processing has been modified to support the Lotus Notes format.
- Calendar: now the Import function processes multiple VCALENDAR objects (required by Mozilla client).
- Calendar: iCalendar composing routines have been redesigned to support various EOL symbols.
- WebAdmin: the Remove Key and Certificate button has been added to the WebUser Prefs page.
- WebUser: additional reply/accept/decline operation cause freebusy updates.
- MAPI: MAPI connector 1.0.71 is included.
- TLS: write operations optimized.
- IMAP: now a RECENT response is sent to the client when RECENT-marked messages are removed from the selected Mailbox.
- DNR: the SRV records (RFC 2782) are supported now.
- RPOP: the "source address" data is recorded now.
- RPOP: the Default IP Address setting is implemented.
- PIPE: the "source address" data is recorded now.
- SMTP: the `.smtpq` Router suffix is implemented.
- CLUSTER: the Cluster Admin Log Level setting is implemented.
- CLUSTER: FTP authentication is implemented via PWD now, no need to enable FTP services on the backends.
- Platform: MS Windows: the Event Log records are properly formatted now.
- Bug Fix: CLI: 4.1.4-4.1.5: the UpdateAccounts command did not work for domain admins.
- Bug Fix: CLI: 4.1.4-4.1.5: the Rename* commands did not understand the domain name in the second parameter.
- Bug Fix: LOCAL: 4.1.4-4.1.5: the "Suspend Message" option did not work correctly on non-cluster systems.
- Bug Fix: WebUser: 4.0-4.1: messages sent encrypted but not signed had incorrect EOLs in the encrypted part.
- Bug Fix: WebUser: 3.5-4.1: mailto: links in messages displayed using WML were replaced with compose.wssp links (instead of wcompose.wssp links).
- Bug Fix: DataSets: 3.4-4.1: under heavy ACAP load sub-datasets could be mishandled, sometimes resulting in server crashes.
- Bug Fix: Foundation: 2.0-4.1: the remove-from-array procedure could access released objects.

**4.1.5** 30-Sep-03

- CLUSTER: HTTPU proxying simplified, upgrades from pre-4.1 versions require full cluster restart.
- Account: 64-bit numbers are used for the mail store size calculations (removing the 2GB limit).
- DNR: the Dummy IP Addresses setting has been added (see the Obscure Settings page).
- HTTP: the HTTP_USER_AGENT data is passed to CGI programs.
- WebUser: now the new File As field value is replacing the old FN VCard attribute, too.
- Rules: the Remember 'From' action does not store more than 500 addresses now (to avoid server overloading).
- WebUser: now the freebusy.vfb file is always stored using CR-LF as the line separator.
- MAPI: MAPI Connector 1.0.68 is included.
- Bug Fix: DIRECTORY: 4.0-4.1: specially [mis]formed DNs could crash the server.
- Bug Fix: Statistics: 4.1b2-b4: the webMail* elements were calculated incorrectly.
- Bug Fix: CLI: 4.1.4: SET[SERVER|CLUSTER]WHITEHOLEIPS commands did not work.
- Bug Fix: SNMP: 3.5-4.1: the snmpInPkts and other elements of this group were reported as COUNTER64 (as opposed to Counter32) elements.

**4.1.4** 20-Sep-03

- CLI: the network CLI processor has been redesigned.
- CLI: the CREATEWEBUSERSESSION operation now updates the LastLogin and LastLoginTime Account-info elements.
- CLI: the CREATEWEBUSERSESSION, GETWEBUSERSESSION, KILLWEBUSERSESSION command parameters and required access rights have been changed.
- LOCAL: the Suspend setting is implemented.
- RPOP: the Use Domain IP Addresses option is implemented.
- RADIUS: CHAP authentication is supported now.
- RADIUS: multi-threaded processing is implemented.
- WSSP: the HASPARENTMAILBOX function and MAILBOXLASTNAME, URLMAILBOXPARENT prefixes are implemented.
- Bug Fix: UTF-8 "BOF tag removing" did not work correctly.
- Bug Fix: WebUser: 4.1.2-4.1.3: ability to refer to arbitrary .wssp pages in sessionless requests disabled the sessionless Personal Site management functions.
- Bug Fix: WebUser: 4.1b-4.1.3: Task composer set the Due Date parameter to the Start Date one.
- Bug Fix: WebUser: 4.1b-4.1.3: storing a Note with a specified original Mailbox, but without original message ID could crash the server.

**4.1.3** 27-Aug-03

- Bug Fix: WebUser: 4.1.2: attempts to open a non-existent Mailbox crashed the server.

**4.1.2** 26-Aug-03

- New Platform: the Linux/AMD-64 (x86-64) version is released.
- CLI: the [GET|SET][SERVER|CLUSTER]INTERCEPT commands are implemented.
- WebAdmin: Skin Editor now removes the "UTF-8 BOM" starting symbols from the uploaded files.
- Intercept: interception of messages sent from the monitored accounts is implemented.
- Intercept: the monitoring type settings are implemented.
- MAPI: the MAPI Connector 1.0.67 is included.
- Account: the "Created" date element is added to account .info data when the account is created.
- GROUPS: now headers added to the original message using the server/cluster-wide Rules are copied to the distributed message.
- LOCAL: direct-mailbox addressing now supports Mailbox aliases (and all variations on the INBOX name).
- WebAdmin: non-ASCII names of Custom and Rename attributes are supported now.
- WSSP: PERIOD-expression optional clause is added to CALENDARTIME, CALENDARLOCALTIME, and DAYTIMEMENU structures.
- WebUser: the CalendarTimePeriod parameter (specifying the steps in the time menus on the Compose page) is added to WebUser Prefs/Settings.
- WebUser: the ListApprove request parameter and ListApproval dataset elements have been added to the Mailbox.wssp and Message.wssp pages.
- Bug Fix: WebUser: 4.1-4.1.1: "bracketed" E-mail addresses in dataset-type AddressBooks were displayed incorrectly.
- Bug Fix: WebUser: 4.1b3-4.1.1: if the "Notes" folder was created automatically, by storing the first note, it did not get the "Notes" Mailbox class assigned.
- Bug Fix: CLUSTER: 4.1b5-4.1.1: the "mail disabled" and some other temp-error codes generated on backend while performing local delivery were interpreted as "flow control" errors.

**4.1.1** 11-Aug-03

- Kernel: the "lawful interception" functionality has been implemented (see the SysAdmin section of the manual).
- GROUP: the Reject Automatic Messages option processing changed to default to "No" for existing groups

created with older versions.

- Kernel: format=flowed (RFC2646) processing has been improved.
- WebUser: AddressBook: Directory books now show the first E-mail address if a record contains several "mail" values.
- WebUser: the "Pty" (X-Priority) header can be used in Mailbox views.
- WSSP: ^y and ^d symbol combinations can be used in Date "pictures".
- WSSP: DAYTIMEMENU, CALENDARTIMEMENU, LOCALCALENDARTIMEMENU elements are documented.
- MAPI: the MAPI Connector 1.0.65 is included.
- WebUser: adding records to Directory-based Address Books is implemented.
- WebUser: if a message has multiple To: and/or multiple Cc: header fields, all their addresses are displayed.
- Skins: the WML files have been modified.
- RPOP: MSN AUTH is used if the host account name ends with "*.msn.com".
- Bug Fix: Directory: 3.5-4.0: non-printable symbols in RDN values were decoded incorrectly.
- Bug Fix: WebUser: 4.1: new cookie processing caused troubles for CGI applications and Cluster environments.
- Bug Fix: WebUser: 4.1b6-4.1: new records created in a Mailbox-type Address Book via the Address Book panel did not show up immediately.

**4.1** 20-Jul-03

- IMAP: the LOGIN-REFERRALS keyword has been added to the CAPABILITY list.
- DataSets: a faster Dictionary file writing method is used.
- CLI: the REJECTQUEUEMESSAGE and GETTEMPCLIENTIPS commands are implemented.
- CLI: the REMOVEACCOUNTSUBSET command is documented.
- RULES: the [RCPT] and [ORCPT] Execute prefixes can be used with Account-level rules.
- RULES: the [ACCNT] prefix is implemented.
- RULES: the ^R macro can be used in Reply and React actions to include the original To: header field.
- RULES: the Submit Address condition is implemented.
- DOMAIN: Directory-based: Domain Rules are implemented.
- LIST: the Moderate Guests posting policy option is implemented.
- HTTP: Cookie processing has been changed to support buggy browsers (Apple's Safari).
- WebUser: messages sent with the Password Recovery component now have a non-empty Return-Path.
- Bug Fix: WebUser: 3.5-4.1b9: when a WebUser Preference setting was set to the "default" value, the old value was not deleted from the dataset file.
- Bug Fix: WebUser: 4.1b8-b9: requests for the contactgroup.wssp page w/o the OrigMailbox parameter could crash the server.
- Bug Fix: Security: 4.1b9: the MD5Crypt routine (used to process $1$-passwords) was broken.
- Bug Fix: WebAdmin: 4.1b1-b9: custom WebAdmin files uploading did not work inside subfolders.
- Bug Fix: DOMAIN: 4.0-4.1b9: Default WebUser Preference could not be created in new Directory-based Domains.
- Bug Fix: WebUser: 4.1b7-b9: server could crash when opening a Mailbox-type Address book if the Mailbox could not be opened.

**4.1b9** 09-Jul-03

- Foundation: non-blocking sockets are implemented (enabled with the --UseNonBlockingSockets option).
- Foundation: the Blowfish encryption cipher is implemented.
- the MAPI Connector 1.0.62 is included (Tasks/ToDo support and other changes).
- RADIUS: the RADIUS server is implemented (See the SysAdmin->RADIUS section of the manual). The "RADIUS" Service has been added to Account and Domain Settings.
- DIRECTORY: now Remote Units can connect to remote LDAP servers securely.
- Skins: tar-archive uploading in implemented.
- Security: the "WebUser" SASL method is implemented.
- Security: the UB-crpt (Unix Blowfish-based) password encryption method is implemented.

- HTTP: the /CalendarData/ realm is implemented to support subscribe/publish HTTP operations for groupware applications.
- LOCAL: the "Delay" settings have been added, the "percent full" processing has been changed to use the current size + new message size in computations.
- IMAP/MAILBOX: internal UID processing has been changed to support unsigned 32-bit numbers.
- FTP: secure (TLS) data connections are supported now.
- WebUser: now Reply/ReplyAll operations use the text/html part of the original message if a text/plain part cannot be found.
- RULES: non-ASCII Mailbox names are now encoded into the "modified UTF-7" charset.
- LIST: non-standard Yahoo bounce reports are processed now.
- ACAP: data strings with binary zeros are now converted to "data" elements, so they can be stored and sent out without string cutting (required for phone numbers in Mulberry address books).
- WSSP: the DATE: and other date/time prefixes now use formatting "pictures". These pictures can be altered using the DatePictures dictionary.
- CLI: the SETMAILBOXCLASS command is implemented.
- Bug Fix: WebUser: 4.1b8: Mailbox-type address books did not display their records.
- Bug Fix: MAILBOX: 4.0-4.1b8: the "Concurrently used large buffers" Obscure setting was reset on server restart.
- Bug Fix: LDAP: 4.0-4.1b8: STARTTLS command was processed incorrectly.
- Bug Fix: WebUser: 4.1b8: requests for the contact.wssp page w/o the OrigMailbox parameter could crash the server.
- Bug Fix: RULES: 4.0-4.1b8: string objects with the ADDHEADER external filter data were not released.

**4.1b8** 12-Jun-03

- WebAdmin: the Clear 'Replied Addresses' List command is added to the Account Rules page.
- WebUser: the Recurrent Events editor is implemented.
- WebUser: Tasks support is implemented.
- WebUser: the Notes editor is implemented.
- WebUser: now the To header field can be included into the Mailbox Viewer field set.
- HTTP: the Request Size Limit setting is implemented (WebAdmin:Settings->WebUser).
- HTTP: the Default Page Name setting is implemented (WebAdmin:Settings->WebUser).
- WSSP: the MAILBOXMENU processing has been changed from a enum-type menu to a string-type menu.
- Kernel: the Euro sign has been added to ISO-8859-x charsets.
- PWD: the STLS command (similar to POP3 STLS command) has been implemented.
- PWD: the challenge-response SASL methods are supported now.
- HTTP: now the HTTP_COOKIE envir-variable is passed to CGI scripts.
- Account: the WebCal Service setting is implemented. This Service controls availability of the WebUser Calendaring functions.
- RULES: the Accept Request action is implemented. It allows to set up accounts for shared resources like conference rooms and schedule them automatically.
- RULES: the Enable Vacation button is supported now (it enables the Vacation Rule and clears the Reply Addresses dataset).
- LIST: headers and footers are converted into HTML if they are inserted into an HTML text.
- LDAP, Directory: the "greaterOrEqual" and "lessOrEqual" operations are implemented.
- HTTP: "local part" Routing result is processed now, allowing an administrator to reroute Domain-level URLs to subdirectories inside Personal Web Sites.
- Platform: a new method for local IP Addresses discovery is implemented.
- Kernel: if a message file with bare-LF gets into the Queue on a 2-byte-EOL platform (Win, OS/2), bare-LFs are not sent out.
- MAILBOX: TextMailbox Manager now uses the fsync() call for all Mailbox update operations. This is required in Dynamic Clusters based on certain Cluster File Systems.
- Bug Fix: WebUser: 4.1b4-b7: All-Day events were processed incorrectly.
- Bug Fix: WebUser: 4.1b7: When an "own event" w/o attendees was removed from the Calendar, the server

could crash.

- Bug Fix: WebUser: 4.1b4-4.1b7: MDN messages did not have the "To" header field.

**4.1b7** 02-Jun-03

- IMAP: RFC3516 (Binary Content Extension) is implemented.
- CLI: the DELETEDIRECTORYRECORDS and INSERTDIRECTORYRECORDS commands are implemented.
- WSSP: LOCALCALENDARDATE and LOCALCALENDARTIME constructs are implemented.
- LIST: if the "notify owner" option is switched on, now the owner gets a message if a subscription was terminated due to confirmation time-out.
- MAPI: the MAPI Connector 1.0.59 is included.
- Directory: Local Units: "top" entries (with empty DNs) can be created and removed.
- Migration: MoveIMAPMail: the --copyMailboxClass option is implemented.
- Migration/Security: the {LANM} and {MSNT} passwords (LAN Manager/Microsoft NT password hashes) are supported now (to simplify migration from Microsoft servers).
- External Programs: the QUIT command has been implemented.
- External AUTH: the ROUTED and FAILURE responses to the NEW command are supported now.
- WebUser: now the TakeAddress, TakeCertificate operations are applied to the currently selected address book.
- WebUser: the Export Calendar Data operation is implemented.
- WebUser: the Export vCard Data operation is implemented.
- WebUser: the Cancel operation is implemented.
- WebUser: S/MIME: various encryption methods are implemented. The Encryption Method settings has been added to the WebUser Settings.
- LIST: several routines have been changed to tie the owner account to the Cluster controller, and to support the owner renaming operation in a Cluster.
- CLI: the GETCURRENTCONTROLLER command is implemented.
- GROUP: the Reject Automatic Messages option is implemented.
- MAILBOX: the anyone@*domainName* special ACL name is implemented to allow users to grant Mailbox access rights to all accounts in the specified Domain.
- Kernel: now MIME parser accepts empty lines in embedded uuencoded files.
- Bug Fix: WebUser: 4.1b4-4.1b6: composed All-Day events could be stored with incorrect dates.
- Bug Fix: WebUser: 4.1b1-4.1b6: browsing a Calendar with events extending outside the "visible window" could crash the server.

**4.1b6** 07-May-03

- WebUser: the Contact Editor is implemented.
- WebUser: the Contact Group Editor is implemented.
- WebUser: Address Books support Groups now.
- Kernel: processing of ORCPT parameters has been changed. It should fix the problem with the new [RCPT] prefix processing introduced in 4.1b5.
- WebUser: multiple Directory-type Address Books are implemented.
- WebUser: the WebUser settings now include Contacts settings.
- WebUser: vCard import is implemented (the mailboxsettings.wssp page).
- HTTPU: if the Personal Web Site "freebusy.vfb" file does not exist, the module creates a dummy FreeBusy dataset and returns it to the client.
- WebUser: the "in frame" option for viewing HTML message parts is implemented via the htmlpart.wssp file.
- Bug Fix: WebUser: 4.1b5: the effective "Default Calendar Mailbox" setting value was taken from the "Save Sent" setting.
- Bug Fix: RULES: 4.1b5: Server-wide Reply and Redirect operations could crash the server.
- Bug Fix: WebUser: 4.1b5: the CalendarEvent parser was broken.
- Bug Fix: 3.0-4.0: Malformed URLs for message parts could crash the server.

**4.1b5** 30-Apr-03

- SMTP: the SMTP input module is redesigned.
- CLI: the GETEFFECTIVEWEBUSER command is implemented.
- MAPI: the MAPI Connector 1.0.57 is included.
- Platform: Solaris: the default "application directory" has been moved from "/usr/local/sbin" to "/opt".
- Account Manager: the listMailboxes functions returns Mailbox aliases, even if the search pattern is specified as "~myaccount/...".
- Kernel: now From:/Sender: addresses for auto-generated messages have the "RealName" <address> format.
- MAILBOX: the `anyone@` special ACL name is implemented to allow users to grant Mailbox access rights to all accounts in their own Domains.
- WebUser: additional "sanity" checks for HTML code were added to disable <script> elements in malformed HTML data.
- WebUser: the Trash and Default Calendar Mailboxes can be selected now (for example, the "Deleted Items" Mailbox can be used as the WebUser "Trash").
- WebUser: the "Address Book" panel has been redesigned to provide access to "ACAP dataset", Directory, and Contact-type Mailboxes. The Filter functionality is added.
- Rules: now the EXECUTE action passes the headers added by other rules and actions to the external task.
- PIPE: the headers added by Server-wide rules are passed to the external task.
- TLS: session caching scheme has been redesigned. Old scheme was not properly releasing unused TLS session data under heavy TLS load.
- Foundation: the PKCS7 "EnvelopedData" objects with non-constructed encrypted data elements are supported now.
- MAILBOX: CLUSTER: now the virtual Mailbox manager does not send empty UID ERASE commands.
- CLUSTER: new log settings have been added.
- Bug Fix: WebUser: 4.1b4: the MDNs were generated if an MDN request header field was found in an embedded message (for example, inside DSN reports).
- Bug Fix: WebUser: 4.1b1-b4: the Compose.wssp page ignored the initial To,Cc,Bcc,Subject URL parameters.
- Bug Fix: WebUser: 4.1b1-b4: dates shown on the freebusy.wssp and calendar.wssp pages were shifted by one if the server time zone was ahead of GMT.
- Bug Fix: WSSP: 4.1b1-b4: the CALENDARDATEMENU construct incorrectly formed non-ASCII elements.

**4.1b4** 18-Apr-03

- WebUser: now when users are disconnected from the server, they can login again and resume their work. The "disconnect.wssp" page is removed, the "login.wssp" page has been modified.
- Kernel: the MDN generator is implemented.
- WebUser: the MDN setting (Never, Manual, Auto) is implemented.
- WebUser: the DisableIPWatch and DisableUseCookie options are added to the login.wssp page.
- The All-Domain Aliases settings are moved from the Local module settings to the Router settings.
- CLI: The GETROUTERSETTINGS and SETROUTERSETTINGS commands are implemented.
- QUEUE: the Outgoing Flow Control option is implemented (to block your own spammer users).
- Foundation: OS/2: timezone processing has been changed.
- Directory Integration: now the "cn" and "sn" attributes are generated as non-empty strings, to allow integration with OpenLDAP and other LDAP servers.
- Events: now the special symbol combination "^2" can be used to include the actual SNMP element value into the event messages.
- CLI: the [GET|SET][SERVER|CLUSTER]WebUserDefaults commands are documented.
- Bug Fix: WebUser: 4.1b3: opening the Mailbox Management page could crash the server.
- Bug Fix: Calendar: normalizing empty FreeBusy data could crash the server.
- Bug Fix: IMAP: when searching foreign accounts with "special" INBOXes, the INBOX name was returned without the "foreign account" prefix.
- Bug Fix: WebUser: S/MIME: 4.1b1-4.1b3: signed letters on platforms with one-symbol EOLs (Unix) were composed incorrectly.
- Bug Fix: Directory: The Remote Unit manager incorrectly encoded the trivial "TRUE" and "FALSE" search conditions.

**4.1b3** 11-Apr-03

- Kernel: VCard objects are supported now (RFC2425, RFC VCard 2.1).
- Kernel: algorithms used in "picture" comparison routines ("*"-wildcarded operations) have been improved.
- Kernel: Calendar: the old (VCal 1.0) format is supported now.
- WebUser: new calendaring options on the compose.wssp page.
- WebUser: the mailbox.wssp processor now adds the "notText" element to messages with Content-Type different from "text/*" (so the "attachment" icon can be displayed).
- WebUser: the settings.wssp page can be divided into several pages (settings.wssp, settings1.wssp, settings2.wssp, settings3.wssp), all with the same processing code.
- WebUser: calendar settings are implemented.
- WebUser: mailboxSettings.wssp: vCal/iCal data import is implemented.
- WebUser: now the Free-Busy information is updated automatically.
- HTTP: the Personal WebSite management page is renamed from Index.html into Index.wssp.
- HTTP: the "freebusy.wssp" Personal WebSite page processor has been implemented. It can be accessed as http://server:8100/~user/freebusy.wssp .
- Migration: the MoveIMAPMail command now copies Mailbox ACLs (if supported).
- Kernel: on some platforms (BeOS) the OS could return an empty system name, causing "Internal Router Loop"s.
- DEQUEUER: the "Warning" message text can be customized now.
- CLI: the ROUTE command now has an optional "MAIL" parameter.
- Bug Fix: 4.1b2: WebAdmin: some "Domains" realm links were not working.
- Bug Fix: 4.1b2: WebAdmin: the Template page crashed the server if the Account Template did not have any "Initial Mailboxes".
- Bug Fix: PIPE: 3.4-4.0x: if a submitted file had the Envelope-ID header field, the message header was corrupted.

**4.1b2** 26-Mar-03

- WebUser: when the Alerts page is displayed instead of some other page, the Alerts page now returns back to that page.
- Protection: a new option is added to specify if a special Header field should be added to messages from blacklisted addresses, or if those messages should be rejected.
- SMTP: the "temporary blacklist" is implemented.
- SMTP: the Disconnect Sender options are implemented.
- RULES: the Header Field condition can use the "in" and "not in" operations.
- RULES: the Domain-wide Rules are implemented.
- RULES: the Source Rule condition is implemented (and documented).
- CLI: GETDOMAINRULES and SETDOMAINRULES commands are implemented.
- WebUser: the "generic mailbox" Web Application code now detects the X-Color message header fields and places them into the "color" element of message dictionaries. This feature can be used to highlight certain messages in Mailboxes.
- WebAdmin: now the Account Template page allows you to specify "classes" for automatically created Mailboxes.
- WebAdmin: the Account Template and Account Default page references have been changed.
- ACCOUNT: now when an Account is being renamed or removed, all WebUser sessions with that account are closed first.
- WebUser: the "Use Cookies" option (WebUser Preferences setting) is implemented.
- MAPI: the 1.0.53 version of the MAPI Connector is included.
- Bug Fix: 3.5x-4.0x: WSSP: the ISINDEX function call caused parser errors.
- Bug Fix: 3.1x-4.0x: ACAP: responses for multi-value entries contained excessive space symbols.
- Bug Fix: 4.1b1: WebUser: an attempt to store an Event with an empty Summary field could crash the server.
- Bug Fix: 4.1b1: WebAdmin: the AccessRights.html page code incorrectly processed account names.
- Bug Fix: 4.1b1: RPOP: the "Leave" record parameter value could be displayed incorrectly.

**4.1b1** 16-Mar-03

- The old (pre-Skins) WebUser Interface has been phased out.
- WebUser: now the "Edit Draft" operation processes attachments in the draft message.
- WebUser: the Calendar.wssp page is implemented.
- WebUser: Mailboxes.wssp: the "folder class" menu is implemented.
- WebUser: the calendarpart.wssp message part viewer page is implemented.
- WebUser: Message.wssp: the Calendar Event operations (Accept/Decline, Update Calendar, Remove from Calendar) are implemented.
- WebUser: the compose.wssp page can create and update Events.
- WebUser: HTML->WML message body translation is implemented.
- WebAdmin: the General->Spelling page with spell checker settings is added.
- WebUser: spellchecker functions have been added to the compose.wssp page.
- SMTP: the Default IP Address setting is implemented.
- WSSP: the "ISEVEN" function is implemented.
- Foundation: Unix: now the fsync()/fdatasync() call is used to flush "rewritten" files to disk.
- LDAP: the "LDAP" 'service' is added to the set of Account/Domain services.
- CLUSTER: LDAP: the account-based BIND operation can now be used in a Cluster.
- MAILBOX: renaming of Mailbox X into X/Y is allowed again, if the rename operation does not involve submailboxes (renaming mailing list Mailboxes into list archives).
- SNMP: "trafficIn" and "trafficOut" elements have been added, the RPOP and LDAP elements have been added.
- MAPI: the 1.0.51 version of the MAPI connector is included.
- Bug Fix: CLUSTER: IMAP/ACAP: login operation failed if the password contained non-printable symbols or symbols '\' and/or '"'.
- Bug Fix: 4.0-4.0.6: RPOP: users could create new RPOP records with poll period smaller than the specified limit.
- Bug Fix: 4.0-4.0.6: when deleting a Domain, the server tried to re-create an Index.data file after the Domain directory has been removed.

# Summary

The old (pre-Skins) WebUser Interface has been phased out.

**Platforms**

- Linux/x86_64 version is released.
- Windows: a new method for local IP Addresses discovery is implemented.
- Solaris: the default "application directory" has been moved from "/usr/local/sbin" to "/opt".
- Unix: now the fsync()/fdatasync() call is used to flush "rewritten" files to disk.
- OS/2: timezone processing has been changed.

**Foundation**

- Non-blocking sockets are implemented (enabled with the --UseNonBlockingSockets option).
- The PKCS7 "EnvelopedData" objects with non-constructed encrypted data elements are supported now.
- The Blowfish encryption cipher is implemented.

**Security**

- The Lawful Intercept functionality is implemented.
- The "WebUser" SASL method is implemented.
- The UB-crpt (Unix Blowfish-based) password encryption method is implemented.

The {LANM} and {MSNT} passwords (LAN Manager/Microsoft NT password hashes) are supported now (to simplify migration from Microsoft servers).

**RADIUS**

- RADIUS server is implemented.

**MAPI**

- MAPI Connector is updated.

**WebUser**

- The Groupware (Calendar,Tasks,Contacts,Notes. Free/Busy) functions are implemented.
- Spellchecker functions have been added to the compose.wssp page.
- The "Use Cookies" option (WebUser Preferences setting) is implemented.
- When users are disconnected from the server, they can login again and resume their work. The "disconnect.wssp" page is removed, the "login.wssp" page has been modified.
- The "Address Book" panel has been redesigned to provide access to "ACAP dataset", Directory, and Contact-type Mailboxes. The Filter functionality is added.
- Address Books support Groups now.
- HTML->WML message body translation is implemented.
- When the Alerts page is displayed instead of some other page, the Alerts page now returns back to that page.
- The "generic mailbox" Web Application code now detects the X-Color message header fields and places them into the "color" element of message dictionaries. This feature can be used to highlight certain messages in Mailboxes.
- The "Edit Draft" operation processes attachments in the draft message.
- Mailbox.wssp: the processor now adds the "notText" element to messages with Content-Type different from "text/*" (so the "attachment" icon can be displayed).
- The settings.wssp page can be divided into several pages (settings.wssp, settings1.wssp, settings2.wssp, settings3.wssp), all with the same processing code.
- The MDN setting (Never, Manual, Auto) is implemented.
- The DisableIPWatch and DisableUseCookie options are added to the login.wssp page.
- Multiple Directory-type Address Books are implemented.
- Additional "sanity" checks for HTML code were added to disable <script> elements in malformed HTML data.
- The Trash and other Default Mailboxes can be selected now (for example, the "Deleted Items" Mailbox can be used as the WebUser "Trash").
- The "in frame" option for viewing HTML message parts is implemented via the `htmlpart.wssp` file.
- The TakeAddress, TakeCertificate operations are applied to the currently selected address book.
- S/MIME: various encryption methods are implemented. The Encryption Method settings has been added to the WebUser Settings.
- The `To` header field can be included into the Mailbox Viewer field set.
- Messages sent with the Password Recovery component now have a non-empty Return-Path.
- Reply/ReplyAll operations use the text/html part of the original message if a text/plain part cannot be found.

**WSSP**

- The "ISEVEN" function is implemented.
- The LOCALCALENDARDATE and LOCALCALENDARTIME constructs are implemented.
- The DATE: and other date/time prefixes now use formatting "pictures". These pictures can be altered using the DatePictures dictionary.
- The MAILBOXMENU processing has been changed from a enum-type menu to a string-type menu.

**Kernel**

- The Euro sign has been added to ISO-8859-x charsets.
- On some platforms (BeOS) the OS could return an empty system name, causing "Internal Router Loop"s.
- The MDN generator is implemented.
- VCard objects are supported now (RFC2425, RFC VCard 2.1).
- Algorithms used in "picture" comparison routines ("*"-wildcarded operations) have been improved.
- Calendar: the old (VCal 1.0) format is supported now.
- MIME parser accepts empty lines in embedded uuencoded files.
- Processing of ORCPT parameters has been changed.
- From:/Sender: addresses for auto-generated messages have the "RealName" <address> format.
- If a message file with bare-LF gets into the Queue on a 2-byte-EOL platform (Win, OS/2), bare-LFs are not sent out.
- DataSets: a faster Dictionary file writing method is used.

## IMAP

- RFC3516 (Binary Content Extension) is implemented.
- Internal UID processing has been changed to support unsigned 32-bit numbers.
- The LOGIN-REFERRALS keyword has been added to the CAPABILITY list.

## PWD

- The STLS command (similar to POP3 STLS command) has been implemented.
- The challenge-response SASL methods are supported now.

## SMTP

- The Default IP Address setting is implemented.
- The "temporary blacklist" is implemented.
- The Disconnect Sender options are implemented.
- The SMTP input module is redesigned.

## LOCAL

- The "Delay" settings have been added, the "percent full" processing has been changed to use the current size + new message size in computations.

## PIPE

- The headers added by Server-wide rules are passed to the external task.

## ACAP

- Data strings with binary zeros are now converted to "data" elements, so they can be stored and sent out without string cutting (required for phone numbers in Mulberry address books).

## HTTP

- The Personal WebSite management page is renamed from Index.html into Index.wssp.
- The "freebusy.wssp" Personal WebSite page processor has been implemented. It can be accessed as http://server:8100/~user/freebusy.wssp .
- If the Personal Web Site "freebusy.vfb" file does not exist, the module creates a dummy FreeBusy dataset and returns it to the client.
- The Request Size Limit setting is implemented (WebAdmin:Settings->WebUser).
- The Default Page Name setting is implemented (WebAdmin:Settings->WebUser).
- The "local part" Routing result is processed now, allowing an administrator to reroute Domain-level URLs to

subdirectories inside Personal Web Sites.

- The HTTP_COOKIE envir-variable is passed to CGI scripts.
- The /CalendarData/ realm is implemented to support subscribe/publish HTTP operations for groupware applications.

## FTP

- Secure (TLS) data connections are supported now.

## LIST

- Several routines have been changed to tie the owner account to the Cluster controller, and to support the owner renaming operation in a Cluster.
- Headers and footers are converted into HTML if they are inserted into an HTML text.
- Non-standard Yahoo bounce reports are processed now.
- The Moderate Guests posting policy option is implemented.
- With the "notify owner" option switched on, the owner gets a message if a subscription was terminated due to confirmation time-out.

## GROUP

- The Reject Automatic Messages option is implemented.

## RULES

- The Domain-wide Rules are implemented.
- The Header Field condition can use the "in" and "not in" operations.
- The Source Rule condition is implemented (and documented).
- The EXECUTE action passes the headers added by other rules and actions to the external task.
- The Accept Request action is implemented. It allows to set up accounts for shared resources like conference rooms and schedule them automatically.
- The Enable Vacation button is supported now (it enables the Vacation Rule and clears the Reply Addresses dataset).
- The [RCPT] and [ORCPT] Execute prefixes can be used with Account-level rules.
- The [ACCNT] prefix is implemented.
- The ^R macro can be used in Reply and React actions to include the original To: header field.
- The Submit Address condition is implemented.
- Non-ASCII Mailbox names are now encoded into the "modified UTF-7" charset.

## CLI

- The GETDOMAINRULES and SETDOMAINRULES commands are implemented.
- The ROUTE command now has an optional "MAIL" parameter.
- The GETROUTERSETTINGS and SETROUTERSETTINGS commands are implemented.
- The [GET|SET][SERVER|CLUSTER]WebUserDefaults commands are documented.
- The GETEFFECTIVEWEBUSER command is implemented.
- The GETCURRENTCONTROLLER command is implemented.
- The SETMAILBOXCLASS command is implemented.
- The DELETEDIRECTORYRECORDS and INSERTDIRECTORYRECORDS commands are implemented.
- The REJECTQUEUEMESSAGE and GETTEMPCLIENTIPS commands are implemented.
- The REMOVEACCOUNTSUBSET command is documented.

## ACCOUNT, DOMAIN

- When an Account is being renamed or removed, all WebUser sessions with that account are closed first.

- The `WebCal` Service setting is implemented. This Service controls availability of the WebUser Calendaring functions.
- The "LDAP" 'service' is added to the set of Account/Domain services.
- The listMailboxes functions returns Mailbox aliases, even if the search pattern is specified as "~myaccount/...".
- Directory-based Domains: Domain Rules are implemented.

**Directory Integration**

- The "cn" and "sn" attributes are generated as non-empty strings, to allow integration with OpenLDAP and other LDAP servers.

**LDAP, Directory**

- Remote Units can connect to remote LDAP servers securely.
- Local Units: "top" entries (with empty DNs) can be created and removed.
- The "greaterOrEqual" and "lessOrEqual" operations are implemented.

**External Programs**

- The QUIT command has been implemented.
- AUTH: the ROUTED and FAILURE responses to the NEW command are supported now.

**TLS**

- Session caching scheme has been redesigned. Old scheme was not properly releasing unused TLS session data under heavy TLS load.

**Events**

- The special symbol combination "^2" can be used to include the actual SNMP element value into the event messages.

**ROUTER**

- The All-Domain Aliases settings are moved from the Local module settings to the Router settings.

**QUEUE**

- The Outgoing Flow Control option is implemented (to block your own spammer users).
- The "Warning" message text can be customized now.

**Protection**

- An option is added to specify if a special Header field should be added to messages from blacklisted addresses, or if those messages should be rejected.

**WebAdmin**

- The General->Spelling page with spell checker settings is added.
- The Account Template page allows you to specify "classes" for automatically created Mailboxes.
- The Account Template and Account Default page references have been changed.
- The Clear 'Replied Addresses' List command is added to the Account Rules page.
- Skins: tar-archive uploading in implemented.

**MAILBOX**

- Renaming of Mailbox X into X/Y is allowed again, if the rename operation does not involve submailboxes (renaming mailing list Mailboxes into list archives).
- The `anyone@` special ACL name is implemented to allow users to grant Mailbox access rights to all accounts in their own Domains.
- The `anyone@domainName` special ACL name is implemented to allow users to grant Mailbox access rights to all accounts in the specified Domain.
- The TextMailbox Manager now uses the fsync() call for all Mailbox update operations. This is required in Dynamic Clusters based on certain Cluster File Systems.

## SNMP

- SNMP: "trafficIn" and "trafficOut" elements have been added, the RPOP and LDAP elements have been added.

## CLUSTER

- LDAP: the account-based BIND operation can now be used in a Cluster.
- The virtual Mailbox manager does not send empty UID ERASE commands.
- New log settings have been added.

## Migration

- The MoveIMAPMail command now copies Mailbox ACLs (if supported).
- MoveIMAPMail: the --copyMailboxClass option is implemented.

## Bug Fixes

- Bug Fix: CLUSTER: IMAP/ACAP: login operation failed if the password contained non-printable symbols or symbols '\' and/or '"'.
- Bug Fix: 4.0-4.0.6: RPOP: users could create new RPOP records with poll period smaller than the specified limit.
- Bug Fix: 4.0-4.0.6: when deleting a Domain, the server tried to re-create an Index.data file after the Domain directory has been removed.
- Bug Fix: 3.5x-4.0x: WSSP: the ISINDEX function call caused parser errors.
- Bug Fix: 3.1x-4.0x: ACAP: responses for multi-value entries contained excessive space symbols.
- Bug Fix: PIPE: 3.4-4.0x: if a submitted file had the Envelope-ID header field, the message header was corrupted.
- Bug Fix: IMAP: when searching foreign accounts with "special" INBOXes, the INBOX name was returned without the "foreign account" prefix.
- Bug Fix: Directory: The Remote Unit manager incorrectly encoded the trivial "TRUE" and "FALSE" search conditions.
- Bug Fix: 3.0-4.0: Malformed URLs for message parts could crash the server.
- Bug Fix: MAILBOX: 4.0-4.1b8: the "Concurrently used large buffers" Obscure setting was reset on server restart.
- Bug Fix: LDAP: 4.0-4.1b8: STARTTLS command was processed incorrectly.
- Bug Fix: RULES: 4.0-4.1b8: string objects with the ADDHEADER external filter data were not released.
- Bug Fix: WebUser: 3.5-4.1b9: when a WebUser Preference setting was set to the "default" value, the old value was not deleted from the dataset file.
- Bug Fix: DOMAIN: 4.0-4.1b9: Default WebUser Preference could not be created in new Directory-based Domains.

# Version 4.0 Revision History

- 6.2
- 6.1
- 6.0
- 5.4
- 5.3
- 5.2
- 5.1
- 5.0
- 4.3
- 4.2
- 4.1
- 4.0: Summary
- 3.x
- 2.x
- 1.x

RSS

**4.0.6** 09-Feb-03

- External AUTH: now the INTF command is always sent to the application as soon as the application is started.
- WebUser: WML support has been extended. WML code has been placed into separate wlogin.wssp, whello.wssp, wbye.wssp, wmailboxes.wssp, wmailbox.wssp, and wmessage.wssp Skin files.
- Directory: Suffix manipulation for "mounted Units" has been changed to support non-canonized DNs.
- Directory: Local Units: now RDNs are not converted into lowercase before storing into the database.
- MAPI: off-line mode supported, the Deleted Items Mailbox is always mapped onto the user own `Deleted Items` Mailbox.
- Bug Fix: de-synching could occur when releasing a dataset file (such as an address book), causing server crashes.
- Bug Fix: WebADMIN: an EXTFILTER could be renamed with a trailing space in the new name, and then that filter could not be removed.
- Bug Fix: the sendmail program could loop when renaming a .tmp file into a .sub file.

**4.0.5** 10-Jan-03

- Rules, PIPE: the addresses added to the command line with the [RETPATH] and [RCPT] headers are enclosed in double quotes now.
- WebUser: WML-related sections were added to .wssp files and wssp processors.
- Bug Fix: Rules: 4.0.4: the charset verification utility could enter an infinite loop.

**4.0.4** 08-Jan-03

- HTTP: CGI: the HTTPS environment variable (with the value "on") is added if the connection is made via the SSL/TLS protocol.
- Directory: RDNs containing the quote mark and comma symbols are supported in Local Units now.
- WSSP: the HTMLUTF8: prefix is implemented.
- WebAdmin: now domain administrators can login using non-qualified aliases.
- MAPI: MAPI Connection v1.0.43 is released.
- Bug Fix: HTTP: CGI: if the port number was not specified in the URL coming via an HTTPS connection, the SERVER_PORT envir-element was set to "80" (instead of "443").
- Bug Fix: Skins: the "stock" Rules.wssp page did not contain the value "9" in the Rule Priority menu.
- Bug Fix: LIST: 4.0b-4.0.3: moderated subscriptions did not work (because the X-LIST-Report: header was added to pending requests).
- Bug Fix: Admin: 4.0.3: the Reject Queued Message function could crash the server if the message was enqueued into several queues.
- Bug Fix: WebUser: 4.0b2-4.0.3: viewing a letter with a zero-length text/plain part could crash the server.
- Bug Fix: WebUser: clicking Update on the Rules.wssp page removed rules with non-ASCII names if the page was not displayed in the UTF-8 charset.
- Bug Fix: WebUser: S/MIME: importing a PKS file w/o a signature could crash the server.
- Bug Fix: External Filter: the server reported its supported Interface version as "1", while it should be "2".

**4.0.3** 16-Dec-02

- WebAmin: the Reject Queued Message function is implemented.
- IMAP: the nesting depth of SEARCH expressions is now limited (to 30) to avoid stack overflows when processing extremely complex SEARCH expressions.
- IMAP: Mailbox access mode for the FETCH operations has changed to avoid resource locking when an IMAP connection is broken.
- Bug Fix: WebAdmin: 4.0.2: the Domain List limit and Object List limit settings always had the same value.
- Bug Fix: WebAdmin: 4.0-4.0.1: the Domain Admin entrance page displayed only one "Other Domain".
- Bug Fix: CLI: 4.0-4.0.2: the GETSUBSCRIBERINFO command did not return the "mode" data.
- Bug Fix: SECURITY: 4.0b-4.0.2: WebUser Interface: some file-path check routines incorrectly checked for the ".." path elements thus providing access to files outside the target directory.

**4.0.2** 26-Nov-02

- Security: the MSN SASL method is implemented.
- WebAdmin: the Admin Preferences for the Domain List page are implemented.
- WebUser: RC2-CBC is now used as the default S/MIME encryption cipher (to support older versions of Netscape).
- Charsets: ISO-8859-10 and ISO-8859-15 charsets are supported now.
- MAPI: the basic "off-line mode" is implemented.
- Bug Fix: WebUser: 4.0-4.0.1: the webSiteEnabled session element was not set if the WebSite quota was set to "unlimited".
- Bug Fix: WebAdmin: 4.0b4-4.0.1: the Refresh Rate for WebUser Sessions monitor could not be set via WebAdmin.
- Bug Fix: WebUser: 4.0b8-4.0.1: embedded images were not displayed in the HTML texts inside multipart/related messages.
- Bug Fix: CLUSTER: specifying '*' as the forwarding host (to relay mail via frontends) did not always work (the port number was not set).
- Bug Fix: Log: 1.0-4.0.1: certain "unusual" data elements could cause the Log engine crashes.
- Bug Fix: WebUser: 3.0-4.0.1: the "message open" procedure could cause deadlocks.

**4.0.1** 06-Nov-02

- Foundation: condition locks are re-implemented for all platforms. Old implementation could cause excessive

- thread waiting times on an overloaded system.
- Protection: DNR error codes can be processed with the Blacklisting by DNS option (making it possible to blacklist all hosts that do not have reverse DNS records).
- EventHandlers: the Frequency parameters have been added.
- EventHandlers: now events are recorded in the CommuniGate Pro Log.
- Skins: the DeliveryReport code and the .wssp file have been added.
- Skins: the Notify when Read option has been added to the Compose code and .wssp file.
- Mailbox Manager: the RENAME operation now automatically creates all "outer" Mailbox folders needed to create the target Mailbox.
- MAPI: many changes and bug fixes (see the MAPI status page).
- LIST: mail header composing routines have been changed.
- CLUSTER: now the cluster-wide RFC822 Filters are not applied when a message is transferred between cluster members for local delivery.
- WebAdmin: the internal Domain Administration realm has been changed to "/DomainAdmin/" to fix various URL access problems.
- Bug Fix: S/MIME: WebUser Interface refused to send an encrypted message to a recipient with a longer public key.
- Bug Fix: SNMP: HTTPAdmin module wrote its stat information into the HTTPUser data structures.
- Bug Fix: SNMP: 64-bit COUNTER data in responses had the APP 1 type instead of the APP 6 type.
- Bug Fix: RULES: the Remember From action could crash the server if no parameter was specified.
- Bug Fix: MAILBOXES: in some rare situations concurrent access to a Mailbox could crash the server.
- Bug Fix: DEQUEUER: if a message had a pending request to generate a "delayed warning" report, and the delayed address failed/succeeded at the same time, the server could crash.
- Bug Fix: Rules: ROUTE conditions could crash the server if an incoming message was rejected because it had "too many hops".

**4.0** 18-Oct-02

- CLI: the GETSUBSCRIBERINFO command is implemented.
- LDAP: binary data are supported now.
- LDAP: if the "Substitute with uid in conditions" option is enabled, the "equals to" search operation ignores the "@domain" part of the search string.
- Directory: Remote Units: binary data are supported now.
- Directory Integration: now the userCertificate attribute is automatically stored in the Directory (together with the RealName/cn attribute).
- CLUSTER: extremely large responses that cannot be read by a requesting cluster member are replaced with an error code on the serving member.
- TLS: a workaround for programs that incorrectly send TLS data in the block-cipher mode (such as Exim MTA) is implemented.
- EXTFILTER: the version 2 API is implemented: the ADDHEADER response code is supported now.
- WebMail: Secure MIME certificate and key page has been reimplemented and documented.
- WebAdmin: the Show Aliases option is implemented.
- WebSites: now the `default.html` file is retrieved for all types of "directory links" (http://server/~user/dir1/dir2/).
- WebSkins: several new SESSION dataset elements have been added.
- WebSkins: the `EmptyTrash` HTTP parameter has been renamed into `EmptyTrashNow`; now it is processed for all session requests.
- WebSkins: default charset processing has been changed.
- WebSkins: .wssp pages for stateless requests can (and should) use the %%filesRef%% variable to form file reference URLs.
- WebUser: the "use Letter charset" option is implemented.
- WebUser: the Store Attachments (Files) function is implemented (the message.wssp page has been extended).
- Kernel: Ukranian KOI8-U charset is supported now.
- Statistics: the "Dynamic Cluster requests" SNMP elements have been added.
- WebAdmin: Directory: the Browser can display non-ASCII data now.

- WebAdmin: "disabled" RPOP periods are supported now.
- CLI: the LISTADMINDOMAINS command is implemented.
- MAPI: support for Outlook Rules has been added.
- Bug Fix: LIST: 4.0b9: digest generator did not calculate the collected message size correctly, and could include too few messages in each digest.
- Bug Fix: Domains: 4.0b7-b9: the Index.data file could fail processing the "removed" (tagged with "-") records.
- Bug Fix: Skins: 4.0b9 "message part" application codes incorrectly processed charsets.
- Bug Fix: CLUSTER: the CREATEWEBUSERSESSION, GETWEBUSERSESSION, KILLWEBUSERSESSION CLI commands could fail in a cluster.
- Bug Fix: External Program parameters were parsed with a wrong parser (adjacent quoted strings were concatenated).
- Bug Fix: S/MIME: signatures did not include all attributes required by some mailers.
- Bug Fix: S/MIME: some signatures could not be verified because of an incorrect buffering technique used.
- Bug Fix: S/MIME: signed and encrypted messages larger than 4K were composed incorrectly.
- Bug Fix: images downloaded from Personal WebSites via WebAdmin Interface could be damaged.
- Bug Fix: MAPI: Windows spooler could crash if the CommuniGate Pro MAPI Connector was used.
- Bug Fix: MAPI: embedded/nested Calendaring messages were processed incorrectly.
- Bug Fix: MAPI: some composed headers were not MIME-encoded.
- Bug Fix: MAPI: drag-and-drop text-type attachments did not work correctly.

**4.0b9** 07-Oct-02

- New Platform: OS2/Intel version is released.
- Foundation: many new cryptography routines are implemented.
- Foundation: binary data can now be stored in directories, datasets, and other basic objects.
- Directory: now when the Direct LDAP Provisioning is enabled, BIND operations can use the Account Manager.
- Migration: now the MoveIMAPMail program supports the --byOne parameter.
- Admin: the SMIME Access Mode option is implemented.
- IMAP: message set and UID message set processing has been changed, to treat an M:N range as the N:M range if M > N.
- ENQUEUER: the Hop Counter Limit settings is implemented.
- Cryptography/TLS: block ciphers are supported now.
- Platform: Win32: under Windows NT/200/XP "OS Log" messages are now stored in the "System Event Log".
- Platform: OS/2: external program launching is implemented now.
- LIST: digesting algorithm has been changed to avoid "message dropping" when the number of collected messages is higher than the digest limit.
- WebSkins: message rendering is controlled by .wssp files now.
- WebSkins: the HTMLTRUNCATED prefix is implemented. Mailbox field values are not truncated by the server code itself.
- WebUser: Settings: the SecureMail settings page is implemented. It can be used to import PFX (.pfx) and PKCS12 (.p12) files with personal private keys and certificates.
- WebUser: Message: decryption of S/MIME encrypted messages is implemented.
- WebUser: Message: digital signature verification of S/MIME signed messages is implemented.
- WebUser: Message: the "Take Certificate" function is implemented (you can now add certificates from signatures to the Address Book).
- WebUser: Compose: the "Send Encrypted" option is implemented.
- WebUser: Compose: the "Send Signed" option is implemented (it can be used to send digitally signed letters).
- MAPI: the Windows Registry routines have been changed to avoid the "access denied" problems.
- MAPI: the MYRIGHTS IMAP operation is used to check the user access rights for the selected Mailbox.
- MAPI: counters are implemented for non-IMAP Mailboxes (Outbox).
- MAPI: the \Answered \Flagged \Redirected IMAP flags are supported now.
- MAPI: "voting" letters are sent in the TNEF format now.
- MAPI: messages stored in the "Sent Items" Mailbox are stored with the SENT flag.
- MAPI: the Reply-To field is supported now.

- MAPI: Outlook 2002: when a new Contact is stored, an RTF part is composed, too.
- MAPI: the Sensitivity header field processing has been changed.
- MAPI: Internal EntryID processing has been changed.
- MAPI: "categories" are supported now.
- MAPI: background pictures in HTML-formatted messages are supported now.
- MAPI: Mailbox caching method has been changed.
- MAPI: when a submailbox is moved, the "+" indicator on the parent Mailbox is removed properly.
- Bug Fix: MAPI: multi-line MIME header fields were not generated correctly.
- Bug Fix: MAPI: E-mail address and Display As fields were not stored properly in received Contact items.
- Bug Fix: MAPI: empty Full Names could cause Connector crashes.
- Bug Fix: MAPI: the "Mark All as Read" operation did not reset the unread message counter.
- Bug Fix: MAPI: Notes and Calendar items were not transferred correctly.
- Bug Fix: MAPI: 4.0b8: retrieval of other user free/busy information was broken.
- Bug Fix: SMTP: 4.0b8: the new RFCWriter did not process the "Add Header" data correctly.
- Bug Fix: RPOP: 4.0b8: could crash in the WebAdmin RPOP Monitor.

**4.0b8** 15-Sep-02

- New Platform: QNX/Intel version is released.
- New Platform: FreeBSD/Alpha version is released.
- SMTP, POP: the RFC822 encoder has been redesigned.
- Personal Web Sites: max directory nesting level has been increased to 6.
- WebUser: the PublicInfo page has been implemented.
- WebUser: MIME->HTML converter checks for binary 0 symbols now.
- Rules: the EACH ROUTE and ANY ROUTE condition processing has been changed.
- EXTFILTER: support for multiple External Filters is implemented.
- CLUSTER: now all inter-cluster connections are initiated using the specified "local cluster IP address".
- CLUSTER: failover algorithms have been improved.
- The --closeStuckSockets command line option has been implemented.
- Accounts: passwords in newly created accounts are encrypted using the effective password encryption method.
- Bug Fix: RPOP: account-level records could become "abandoned" and the server stopped polling them.
- Bug Fix: Account: the "oldest message" timestamp calculation routine was not properly used.
- Bug Fix: the Rules.wssp page incorrectly processed "uneditable" rules.
- Bug Fix: Events: Event Elements were not removed when their Threshold values were reset.
- Bug Fix: Events: thresholds for counter-type Events were not calculated correctly.
- Bug Fix: CLUSTER: failure of a cluster member on a heavily loaded cluster could cause a deadlock in the current Cluster controller.

**4.0b7** 22-Aug-02

- New Platform: HPUX/Itanium version is released.
- RULES: "string lists" are implemented. The Auto-Reply Rule has been modified to send only one reply to each sender.
- SMTP: the Recipients/Message setting has been implemented.
- CLUSTER: SMTP: the "*" forward to option (meaning "to all frontends") is supported now.
- Directory: the DNs of found records are now checked against the Access Rights. If the RDN is not allowed to be read, the record is not returned to the client.
- Directory: to comply with RFC2256, the cn and dc attributes have been moved from the organization objectClass to the communiGateDomain objectClass.
- CLI: the SETPOSTINGMODE command is implemented.
- LOCAL: now distribution to all@domain is prohibited for the messages with an empty return-path (to stop the "bounce mail distribution" attacks.
- Admin: the MAPI Connections setting has been added to the Limits panel on the Domain Settings page.
- WebAdmin: now the Server Up-Time is displayed on the Settings -> General page.

- MAPI: if a newly created message does not have a plain-text part, the RTF part is used (these RTF-only messages were sent as empty messages before).
- MAPI: if a message has an HTML part, other parts are "disabled" (these messages were copied as empty messages before).
- MAPI: non-system locales are supported now.
- MIME: strings: the '\' symbol processing has been improved.
- Skins: the default Skin HTML code has been "cleaned".
- RPOP: internal scheduling mechanisms have been re-designed.
- Platform: the MacOS X package now includes the Uninstall.sh script.
- Bug Fix: MAPI: there could be a deadlock in the spooler, causing Outlook to hang when the Send button was pressed.
- Bug Fix: MAPI: ITable interface bugs could crash Outlook.
- Bug Fix: DOMAINS: the Index.data files have been processed incorrectly if there were account names starting with the "-" sign.

**4.0b6** 30-Jul-02

- LDAP: SASL authentication for `mail=accountName` BindDN is implemented.
- Charsets: data encoded using Korean charsets can be converted to and from the UTF-8 charset.
- WSSP: the DATE:, DATETIME:, LOCALDATETIME:, DATETIMESHORT:, LOCALDATETIMESHORT: prefixes are implemented.
- WebApp: the "Mailbox" processing code has been changed, so the Sent and Received column values are now returned as "date"-type elements.
- WebApp: the "Mailboxes" processing code now adds the "parent" element to the Mailbox elements.
- WebAdmin: now administrators without the CanCreateGroup access right cannot modify any group data.
- WebAdmin: GROUP: several member addresses can be entered in one field now.
- WebMail: now moving a message between "own" Mailboxes (including moving a message to the Trash) temporarily disables storage quota checks.
- HTTP: CGI: the SCRIPT_NAME parameter does not contain the "additional path" data now.
- HTTP: CGI: now an external program receives the PATH_INFO and REQUEST_URI parameters.
- IMAP: RFC3348 (CHILDREN extensions) is implemented.
- MAPI: the version-checking schema has been changed.
- MAPI: ReadReceipts are working now.
- MAPI: now messages are always sent via the spooler.
- MAPI: Win95 systems are supported now.
- MAPI: Unseen message number calculation is improved.
- MAPI: non-default text encodings are supported now.
- MAPI: X-Mailer header is inserted into the composed messages.
- MAPI: "simple" calendar requests are stored and transferred in the Outlook-compatible form.
- MAPI: now the EXPUNGE command is sent to the selected Mailbox.
- Platform: the "application directory" for Linux systems has been changed from `/usr/local/sbin` to `/opt` .
- Foundation: the cryptography routines have been optimized to increase SSL/TLS performance.
- Bug Fix: 4.0b: CLUSTER: failure to open an account on a backend server could crash the frontend server.
- Bug Fix: the "copyMessages" Mailbox Manager routine ignored errors when it was copying large (over 1MB) messages followed by a smaller message.
- Bug Fix: the "oldestMessage" info was not updated when a new message was added to a Mailbox.
- Bug Fix: SMTP: 4.0b5 could not open the Listener WebAdmin page.
- Bug Fix: WebUser: the text/enriched converter did not replace single EOL with a white space symbol.
- Bug Fix: MAPI: the "Remember password" setup option did not work correctly.
- Bug Fix: MAPI: the Date: field is now composed correctly, using the current daylight saving time settings.
- Bug Fix: MAPI: generated messages could contain empty bodies.
- Bug Fix: MAPI: RTF EOL processing has been fixed.
- Bug Fix: MAPI: iCalendar data now contains the UID attribute.
- Bug Fix: MAPI: in some situations the Inbox Mailbox was displayed twice.

**4.0b5** 08-Jul-02

- Events are implemented (see the SysAdmin->Events section for more details).
- HTTP: the module has been divided into two modules (HTTP Admin and HTTP User).
- Admin/Directory Integration: the Public Info Custom Settings are implemented.
- Admin: the MaxMailboxes Account Setting is implemented.
- WebAdmin: the Threads Monitor has been implemented.
- WebAdmin: the Access Monitor pages have been re-implemented. LDAP, FTP, ACAP, PWD monitors are implemented.
- SMTP: the actual address of the blacklisted host is now included into the negative (591) SMTP server response.
- SMTP: now the Send Secured and Hold Mail names can include the wildcard (*) symbols.
- SNMP: the "AUTH" group of statistic elements has been implemented.
- WebAdmin: name filtering is implemented on the SNMP Element List Monitor page.
- WebUser: HTTP responses for .wspp requests now have explicitly specified charsets in the Content-Type headers.
- WebAdmin: HTTP responses now have explicitly specified charsets in the Content-Type headers.
- WSSP: now the Auto-signup component code can put Custom Attribute values into the new Account settings.
- FTP: LIST and NLST command parameters (strings staring with the "-" signs) are skipped.
- FTP: the CWD "/" command is allowed now.
- CLUSTER: the account info is now displayed on Account list and Account Settings WebAdmin pages of all Cluster members.
- MAPI: processing has been changed to utilize only one IMAP connection per client.
- MAPI: new Read Receipts (MSN) are generated and sent.
- MAPI: now the Setup program has the Install/Upgrade/Uninstall options.
- MAPI: the Outlook Setup mode (Corporate/Internet) is detected now and the user is warned if the Corporate mode is switched off.
- MAPI: Address Book: apostrophes are removed from recipient names.
- MAPI: now the Contacts info is stored in the standard vCard format, too.
- MAPI: the X-MAPI-Message-Class header is now added to "special-class" message objects.
- MAPI: Free-Busy information processing has been changed to support shared Accounts.
- Bug Fix: EXTERNALFILTER: in 4.0b4: the INTF command replies where not processed correctly.
- Bug Fix: the MIME parser could return negative part size values for malformed MIME messages.
- Bug Fix: MAPI: message header encoding has been fixed.
- Bug Fix: MAPI: quoted-printable message body encoding did not work correctly.

**4.0b4** 18-Jun-02

- Foundation: Windows: OS privileges needed to enable external program termination are enabled explicitly now.
- External Filtering: the API has been extended (the INTF command added, the DISCARD response is supported now).
- Mailbox: the "oldest message" info is now stored in the Mailbox meta-data dictionary.
- MAPI: several minor bugs are fixed.
- Admin: the Rename In Folder option is implemented. The "renameInPlace" attribute has been added to the Directory Schema.
- WebAdmin: Login page for Domain WebAdmin Interfaces now lists "Other Domains" that this Domain Administrators can open.
- Bug Fix: 4.0b3 could crash when checking Mailbox access rights.

**4.0b3** 16-Jun-02

- The Linux/Itanium version is released.
- MAPI Connector is released.
- Admin: the MAPI and TLS Account and Domain "Enabled Service" options are implemented.
- CLUSTER: Virtual Mailbox manager did not read the UID information for appended messages. As a result the

IMAP APPEND and COPY operations applied to virtual Mailboxes did not return the extended UIDPLUS codes.

- Security: the WebAdmin "Security" page now accepts multi-certificate CA Chains.
- Security: the Advertise NTLM SASL Method option has been added to the Obscure Settings page.
- SMTP: the Advertise NTLM AUTH option has been removed from the SMTP module settings.
- Rules: now the Subject and From Name fields are MIME-decoded and converted into UTF-8 before processing.
- WebAdmin: the Server-Wide and Cluster-Wide "default WebAdmin" pages are supported now.
- LIST: now the ^I macro can be used in the Hello and GoodBye messages.
- ALERTS: Account-level Alerts have been implemented.
- CLI: the GETACCOUNTALERTS, SETACCOUNTALERTS, POSTACCOUNTALERT, and REMOVEACCOUNTALERT commands have been implemented.
- CLI: the GETSNMPELEMENT command has been implemented.
- WebUser: now Named Skins can be used in Stateless requests (such as login.wssp requests).
- WebMail: the GBK charset is supported now.
- Bug Fix: CLUSTER: SMTP: successful client authentication did not reset the "blacklisted IP" flag.
- Bug Fix: MAILBOX: the "Recent" flag was not always processed correctly in the concurrent access environments.
- Bug Fix: the Big5->UTF-8 conversion routine did not process all Big5 symbols correctly.

**4.0b2** 02-Jun-02

- CLUSTER: the GetAccountInfo CLI command is "clusterized" now.
- MIME: search for body strings now works for non-ASCII strings specified in various charsets.
- Security: the Advertise secure SASL Methods option has been added to the Obscure Settings page.
- Directory: new Unit names are now checked for "bad symbols".
- IMAP: now the GETQUOTA/GETQUOTAROOT commands can be used to retrieve Mailbox store information for foreign Mailboxes.
- Protection: now "Unblacklisted" (White Hole) addresses can be specified using their DNS Names.
- PIPE: now the module does not try to "fix" multi-line From: and Sender: header fields in submitted messages.
- Security: the External Authentication API has been changed (see the Security section for more details).
- Domains: now the Index.data file is kept in sync with the Domain Accounts (rather than created on server shutdown).
- Domains: now the Index.data file can be stored in the "Index" subdirectory (to simplify symlink processing).
- Foundation: Tru64: the system mode allowing 65000+ file descriptors is enabled now.
- Bug Fix: IMAP: the QUOTA STORAGE responses now specify sizes in 1K units.
- Bug Fix: IMAP: UID-based non-range messagesets were formed incorrectly if a message with the specified UID did not exist.
- Bug Fix: FTP: the SIZE command could crash the server.
- Bug Fix: FTP: the QUIT command was not process correctly after an unsuccessful login attempt.

**4.0b1** 02-May-02

- FTP: the FTP module is implemented. See the Access->FTP section of the manual for details.
- WebMail: the WML requests are supported now.
- MAILBOX: the information about the number of Unseen messages is cached now.
- WebUser: the information about the number of Unseen messages in now available on the mailboxes.wssp page.
- MAILBOX/IMAP: additional extensions are implemented to simplify communication with the MAPI Connector.
- CLUSTER: now users with the Require Secure Login setting enabled can login using secure connections to frontends.
- RULES: the ^S and ^F macros are now implemented as ^S/^s and ^F/^f macros.
- Router: update procedure has been modified to avoid crashes during Router Table updates under heavy load.
- LOCAL: the Always Add the Envelope Addresses Field option is implemented.
- CLI: the RELEASESMTPQUEUE command is implemented.
- CLUSTER: SMTP: ETRN commands and Wake-up E-mails release queues on all Dynamic Cluster members.

- HTTP: the OPTIONS method is implemented.
- WebUser: the text/enriched format is supported now.
- RPOP: the secure (TLS) option is implemented. The WebSkin, WebAdmin, WebUser RPOP files are updated to accommodate the new option.
- Directory: LDAP provisioning now detects the `unixPassword` attribute and stores its value as a U-crpt'ed Password.

# Summary

**Platforms**

- OS2/Intel version is released.
- QNX/Intel version is released.
- FreeBSD/Alpha version is released.
- HPUX/Itanium version is released.
- Linux/Itanium version is released.

**Events**

- Events are implemented.

**MAPI**

- MAPI Connector is released.

**FTP**

- the FTP module is implemented. See the Access->FTP section of the manual for details.

**Security**

- The MSN SASL method is implemented.
- The WebAdmin "Security" page now accepts multi-certificate CA Chains.
- The Advertise NTLM SASL Method option has been added to the Obscure Settings page.
- The External Authentication API has been changed (see the Security section for more details).
- The Advertise secure SASL Methods option has been added to the Obscure Settings page.
- SNMP: the "AUTH" group of statistic elements has been implemented.

**Protection**

- Now "Unblacklisted" (White Hole) addresses can be specified using their DNS Names.
- DNR error codes can be processed with the Blacklisting by DNS option (making it possible to blacklist all hosts that do not have reverse DNS records).

**WebAdmin**

- The Admin Preferences for the Domain List page are implemented.
- The internal Domain Administration realm has been changed to "/DomainAdmin/" to fix various URL access problems.
- The Show Aliases option is implemented.
- Directory: the Browser can display non-ASCII data now.
- "disabled" RPOP periods are supported now.
- Now the Server Up-Time is displayed on the Settings -> General page.

Now administrators without the CanCreateGroup access right cannot modify any group data.

- GROUP: several member addresses can be entered in one field now.
- The Threads Monitor has been implemented.
- The Access Monitor pages have been re-implemented. LDAP, FTP, ACAP, PWD monitors are implemented.
- Name filtering is implemented on the SNMP Element List Monitor page.
- HTTP responses now have explicitly specified charsets in the Content-Type headers.
- Login page for Domain WebAdmin Interfaces now lists "Other Domains" that this Domain Administrators can open.
- The Server-Wide and Cluster-Wide "default WebAdmin" pages are supported now.

## HTTP

- The module has been divided into two modules (HTTP Admin and HTTP User).
- CGI: the SCRIPT_NAME parameter does not contain the "additional path" data now.
- CGI: now an external program receives the PATH_INFO and REQUEST_URI parameters.
- the OPTIONS method is implemented.

## SMTP

- The Recipients/Message setting has been implemented.
- The actual address of the blacklisted host is now included into the negative (591) SMTP server response.
- Now the Send Secured and Hold Mail names can include the wildcard (*) symbols.
- The Advertise NTLM AUTH option has been removed from the SMTP module settings.

## Local Delivery

- Now distribution to all@domain is prohibited for the messages with an empty return-path (to stop the "bounce mail distribution" attacks.
- The Always Add the Envelope Addresses Field option is implemented.

## LIST

- Mail header composing routines have been changed.
- The digesting algorithm has been changed to avoid "message dropping" when the number of collected messages is higher than the digest limit.
- Now the ^I macro can be used in the Hello and GoodBye messages.

## IMAP

- Message set and UID message set processing has been changed, to treat an M:N range as the N:M range if M > N.
- RFC3348 (CHILDREN extensions) is implemented.
- Now the GETQUOTA/GETQUOTAROOT commands can be used to retrieve Mailbox store information for foreign Mailboxes.
- Additional extensions are implemented to simplify communication with the MAPI Connector.

## RPOP

- RPOP: internal scheduling mechanisms have been re-designed.
- RPOP: the secure (TLS) option is implemented. The WebSkin, WebAdmin, WebUser RPOP files are updated to accommodate the new option.

## PIPE

- Now the module does not try to "fix" multi-line From: and Sender: header fields in submitted messages.

## Router

- Router: update procedure has been modified to avoid crashes during Router Table updates under heavy load.

## Rules

- The EACH ROUTE and ANY ROUTE condition processing has been changed.
- "String lists" are implemented. The Auto-Reply Rule has been modified to send only one reply to each sender.
- Now the Subject and From Name fields are MIME-decoded and converted into UTF-8 before processing.
- The ^S and ^F macros are now implemented as ^S/^s and ^F/^f macros.

## External Filtering

- The version 2 API is implemented: the ADDHEADER response code is supported now.
- Support for multiple External Filters is implemented.
- The API has been extended (the INTF command added, the DISCARD response is supported now).

## LDAP

- Binary data are supported now.
- If the "Substitute with uid in conditions" option is enabled, the "equals to" search operation ignores the "@domain" part of the search string.
- SASL authentication for `mail=accountName` BindDN is implemented.
- Now when the Direct LDAP Provisioning is enabled, BIND operations can use the Account Manager.
- LDAP provisioning now detects the `unixPassword` attribute and stores its value as a U-crpt'ed Password.

## Directory

- The DNs of found records are now checked against the Access Rights. If the RDN is not allowed to be read, the record is not returned to the client.
- Remote Units: binary data are supported now.
- New Unit names are now checked for "bad symbols".
- To comply with RFC2256, the cn and dc attributes have been moved from the organization objectClass to the communiGateDomain objectClass.

## Cluster

- Now the cluster-wide RFC822 Filters are not applied when a message is transferred between cluster members for local delivery.
- Extremely large responses that cannot be read by a requesting cluster member are replaced with an error code on the serving member.
- Now all inter-cluster connections are initiated using the specified "local cluster IP address".
- Failover algorithms have been improved.
- SMTP: the "*" forward to option (meaning "to all frontends") is supported now.
- The account info is now displayed on Account list and Account Settings WebAdmin pages of all Cluster members.
- Statistics: the "Dynamic Cluster requests" SNMP elements have been added.
- Virtual Mailbox manager did not read the UID information for appended messages. As a result the IMAP APPEND and COPY operations applied to virtual Mailboxes did not return the extended UIDPLUS codes.
- The GetAccountInfo CLI command is "clusterized" now.
- Now users with the Require Secure Login setting enabled can login using secure connections to frontends.
- SMTP: ETRN commands and Wake-up E-mails release queues on all Dynamic Cluster members.

## Mailbox Manager

- The "oldest message" info is now stored in the Mailbox meta-data dictionary.
- The information about the number of Unseen messages is cached now.
- The RENAME operation now automatically creates all "outer" Mailbox folders needed to create the target Mailbox.
- The MaxMailboxes Account Setting is implemented.

## Account Manager

- The Public Info Custom Settings are implemented.
- The Rename In Folder option is implemented. The "renameInPlace" attribute has been added to the Directory Schema.
- Passwords in newly created accounts are encrypted using the effective password encryption method.
- Now the userCertificate attribute is automatically stored in the Directory (together with the RealName/cn attribute).

## Domain Manager

- The MAPI and TLS Account and Domain "Enabled Service" options are implemented.
- Now the Index.data file is kept in sync with the Domain Accounts (rather than created on server shutdown).
- Now the Index.data file can be stored in the "Index" subdirectory (to simplify symlink processing).

## Alerts

- Account-level Alerts have been implemented.

## CLI

- The GETSUBSCRIBERINFO command is implemented.
- The SETPOSTINGMODE command is implemented.
- The LISTADMINDOMAINS command is implemented.
- The GETACCOUNTALERTS, SETACCOUNTALERTS, POSTACCOUNTALERT, and REMOVEACCOUNTALERT commands have been implemented.
- The GETSNMPELEMENT command has been implemented.
- The RELEASESMTPQUEUE command is implemented.

## Personal Web Sites

- Now the `default.html` file is retrieved for all types of "directory links" (http://server/~user/dir1/dir2/).
- Max directory nesting level has been increased to 6.

## WebUser Interface

- Now Named Skins can be used in Stateless requests (such as login.wssp requests).
- Message rendering is controlled by .wssp files now.
- The WML requests are supported now.
- The "use Letter charset" option is implemented.
- The Auto-signup component code can put Custom Attribute values into the new Account settings.
- The Store Attachments (Files) function is implemented (the message.wssp page has been extended).
- Settings: the SecureMail settings page is implemented. It can be used to import PFX (.pfx) and PKCS12 (.p12) files with personal private keys and certificates.
- Message: decryption of S/MIME encrypted messages is implemented.
- Message: digital signature verification of S/MIME signed messages is implemented.
- Message: the "Take Certificate" function is implemented (you can now add certificates from signatures to the Address Book).
- Compose: the "Send Encrypted" option is implemented.

- Compose: the "Send Signed" option is implemented (it can be used to send digitally signed letters).
- The PublicInfo page has been implemented.
- The MIME->HTML converter checks for binary 0 symbols now.
- Moving a message between "own" Mailboxes (including moving a message to the Trash) temporarily disables storage quota checks.
- HTTP responses for .wspp requests now have explicitly specified charsets in the Content-Type headers.
- The information about the number of Unseen messages in now available on the mailboxes.wssp page.
- The text/enriched format is supported now.
- The DeliveryReport code and the .wssp file have been added.
- The Notify when Read option has been added to the Compose code and .wssp file.
- Several new SESSION dataset elements have been added.
- The `EmptyTrash` HTTP parameter has been renamed into `EmptyTrashNow`; now it is processed for all session requests.
- Default charset processing has been changed.
- .wssp pages for stateless requests can (and should) use the %%filesRef%% variable to form file reference URLs.
- The HTMLTRUNCATED prefix is implemented. Mailbox field values are not truncated by the server code itself.
- the DATE:, DATETIME:, LOCALDATETIME:, DATETIMESHORT:, LOCALDATETIMESHORT: prefixes are implemented.
- The "Mailbox" processing code has been changed, so the Sent and Received column values are now returned as "date"-type elements.
- The "Mailboxes" processing code now adds the "parent" element to the Mailbox elements.

## Foundation

- Condition locks are re-implemented for all platforms. Old implementation could cause excessive thread waiting times on an overloaded system.
- Many new cryptography routines are implemented.
- Binary data can now be stored in directories, datasets, and other basic objects.
- The cryptography routines have been optimized to increase SSL/TLS performance.
- The --closeStuckSockets command line option has been implemented.
- Linux: the "application directory" for Linux systems has been changed from `/usr/local/sbin` to `/opt` .
- Win32: OS privileges needed to enable external program termination are enabled explicitly now.
- Win32: under Windows NT/200/XP "OS Log" messages are now stored in the "System Event Log".
- Tru64: the system mode allowing 65000+ file descriptors is enabled now.

## Miscellaneous

- TLS: block ciphers are supported now.
- ENQUEUER: the Hop Counter Limit settings is implemented.
- SMTP, POP: the RFC822 encoder has been redesigned.
- MIME: Search for body strings now works for non-ASCII strings specified in various charsets.
- Migration: now the MoveIMAPMail program supports the --byOne parameter.

## Charsets

- ISO-8859-10 and ISO-8859-15 charsets are supported now.
- Ukranian KOI8-U charset is supported now.
- data encoded using Korean charsets can be converted to and from the UTF-8 charset.
- the GBK charset is supported now.

## Bug Fixes

- Bug Fix: SNMP: 64-bit COUNTER data in responses had the APP 1 type instead of the APP 6 type.
- Bug Fix: RULES: the Remember From action could crash the server if no parameter was specified.

- Bug Fix: MAILBOXES: in some rare situations concurrent access to a Mailbox could crash the server.
- Bug Fix: DEQUEUER: if a message had a pending request to generate a "delayed warning" report, and the delayed address failed/succeeded at the same time, the server could crash.
- Bug Fix: Rules: ROUTE conditions could crash the server if an incoming message was rejected because it had "too many hops".
- Bug Fix: Log: 1.0-4.0.1: certain "unusual" data elements could cause the Log engine crashes.
- Bug Fix: WebUser: 3.0-4.0.1: the "message open" procedure could cause deadlocks.
- Bug Fix: CLUSTER: the CREATEWEBUSERSESSION, GETWEBUSERSESSION, KILLWEBUSERSESSION CLI commands could fail in a cluster.
- Bug Fix: External Program parameters were parsed with a wrong parser (adjacent quoted strings were concatenated).
- Bug Fix: images downloaded from Personal WebSites via WebAdmin Interface could be damaged.
- Bug Fix: the "copyMessages" Mailbox Manager routine ignored errors when it was copying large (over 1MB) messages followed by a smaller message.
- Bug Fix: WebUser: the text/enriched converter did not replace single EOL with a white space symbol.
- Bug Fix: CLUSTER: failure of a cluster member on a heavily loaded cluster could cause a deadlock in the current Cluster controller.
- Bug Fix: the MIME parser could return negative part size values for malformed MIME messages.
- Bug Fix: IMAP: the QUOTA STORAGE responses now specify sizes in 1K units.
- Bug Fix: CLUSTER: SMTP: successful client authentication did not reset the "blacklisted IP" flag.
- Bug Fix: MAILBOX: the "Recent" flag was not always processed correctly in the concurrent access environments.
- Bug Fix: the Big5->UTF-8 conversion routine did not process all Big5 symbols correctly.

---

# Version 3.x Revision History

- **6.2**
- **6.1**
- **6.0**
- **5.4**
- **5.3**
- **5.2**
- **5.1**
- **5.0**
- **4.3**
- **4.2**
- **4.1**
- **4.0**
- **3.x: Summary**
- **2.x**
- **1.x**

RSS

**3.5.9** 05-Apr-02

- POP: the '#' symbol processing has been changed to allow re-routing of 'user#domain' account names into 'user@domain' names.
- MAILBOXES: MailDir Mailboxes under Windows are now opened in the Shared mode, allowing several clients to read the same message at the same time w/o the --SharedFiles option.
- Bug Fix: MAILBOXES: deleting messages from MailDir Mailboxes used by several clients could crash the server.
- Bug Fix: PIPE: the 3.5.8 version incorrectly enqueued messages directed to the PIPE module.
- Bug Fix: on some platforms timezone switches (such as daylight saving time switches) have not been detected correctly.

**3.5.8** 23-Mar-02

- LIST: now the administrative message subjects can be specified in the List Settings. The Stock Skin listsettings.wssp file has been modified to include these new settings.
- LIST: if a posted message does not have the charset specified, the Feed mode header and trailer are added to the distributed message in the List Preferred charset.
- LIST: newly created lists now have the same preferred charset as the effective WebUser preferred charset for the list owner account.
- Bug Fix: CLUSTER: the HTTP Keep-Alive option enabled on backend servers could cause user disconnects.
- Bug Fix: CLUSTER: SSL connection proxying could pass incorrect data to the server, causing (mostly) IMAP APPEND errors.
- Bug Fix: IMAP: under certain OS'es (AIX) the fetch command could return "syntax error" when it was issued for an empty Mailbox.

- Bug Fix: message batch enqueueing routines could cause crashes during the server shutdown procedure.

**3.5.7** 11-Mar-02

- Protection: the Blacklisting by IP Domain Name feature has been implemented.
- CLI: the LISTSUBSCRIBERS command now needs the FILTER keyword.
- CLUSTER: now cluster members do not disconnect from the backup controller even if the failover transition takes more than 3 minutes.
- CLUSTER: now the LIST and LISTSUBSCRIBERS commands can be used on any cluster member and they can deal with mailing lists in shared domains.
- MIME: MIME header decoder now removes unencoded white spaces between encoded atoms.
- HTTP: optional support for the Keep-Alive method is implemented.
- HTTP: a workaround for Microsoft Internet Explorer browser bug is implemented.
- LIST: now if the posting policy is set the "Moderate all", non-subscribers can submit their posts, too.
- MAILBOXES: now the TextMailbox scanner can scan several large Mailboxes at the same time.
- Bug Fix: 3.5.6: MIME header encoding module did not encode spaces separating encoded atoms.
- Bug Fix: 3.5.3-3.5.6: the "host queue splitting" operation did not properly release error message object and could cause memory leaks.

**3.5.6** 18-Feb-02

- Bug Fix: 3.5.5: Mailbox size check routine could crash the server if a message was being added to an empty Mailbox in the "parsed" state.

**3.5.5** 14-Feb-02
**Note:** In a dynamic cluster, each frontend must be restarted (during or after its upgrade) AFTER all backends are upgraded.

- LIST: settings are stored in the UTF-8 charset now.
- CLUSTER: message object attributes ("trusted source", "auth-ed") are now sent to backends.
- SMTP: low-level logging for incoming connections is improved.
- Bug Fix: WebAdmin: the Server-wide defaults were displayed instead of Cluster-wide defaults for the shared Domain WebUser Prefs defaults.
- Bug Fix: Directory: if no Unit existed, the Browser could crash the server.
- Bug Fix: Skins: skin file uploading can cause crashes on heavily loaded systems.
- Bug Fix: WebUser: the "HTML message cleanup" module could enter a loop blocking a WebUser session and its open Mailbox.

**3.5.4** 03-Feb-02

- HELPERS: time-out and auto-restart settings are implemented, the HELPER object internals have been re-written.
- Skins: file upload algorithms has been changed, now CLI commands clear caches.
- CLI: the VERIFYACCOUNTPASSWORD command is implemented.
- LIST: if a list subscriber is a local account, the account password can be used to browse "subscriber-only" list archives.
- WebAdmin: the Admin Prefs now can specify the default limit for the number of List Subscribers to display.
- SMTP: reversing the channel after ATRN is sent now works even when the connection is secured using TLS/SSL.
- SMTP: ATRN 4xx response codes have been changed to the values specified in RFC2645.
- Bug Fix: SMTP: receiving ATRN over TLS could crash the server.

**3.5.3** 17-Jan-02

- New Platform release: BeOS/PowerPC.

- CLUSTER: algorithms have been improved to better handle situations when an account create/rename/remove operation is being executed at the same time when the account list is being built. For domains with more than 200,000 users these situations could block access to hash tables for several minutes, causing cluster break-ups.
- LOCAL: distribution rights for the `all@` addresses are now checked before the account list is being built.
- LIST: the "listserver" address processing has been changed. The new processing method does work in the Dynamic Cluster environment, too.
- SMTP: the Relay to Client IP Addresses = simple option processing has been changed to check the original addresses, not the resulting, re-routed addresses.
- SNMP: WebAdmin pages now display 64-bit numeric values.
- Bug Fix: LIST: the "cleanup procedure" closed list owner accounts on the Cluster Controller, allowing them to migrate to other cluster members and lose connections with their lists.

**3.5.2** 05-Jan-02

- Bug Fix: IMAP: 3.5b-3.5.1: the CAPABILITY response did not have a space before the first AUTH= parameter.
- Bug Fix: Account: when a Mailbox without submailboxes was removed using 3.5x versions, the Mailbox size was not subtracted from the account total message storage size.
- Bug Fix: McAfee License Limit Expiration was not always calculated correctly.
- Big Fix: LIST: subscription via the "listserver" address could fail because of incorrect confirmation string processing.
- Bug Fix: WebUser: the last symbol of the "From" address was not stored if the MIME Headers option was switched off.
- Bug Fix: IMAP: 3.5x versions did not support the UID EXPUNGE command.
- Bug Fix: CLI: 3.5x versions could return improperly terminated strings in GetAccountLocation responses.
- Bug Fix: Directory-Based Domains did not "see" their domain Skins after a server restart.

**3.5.1** 18-Dec-01

- WebUser: the Expire header with negative date (added in the 3.5 version) has been removed, since it caused problems for old Netscape browsers.
- Lists: the Dynamic Cluster Controller now tries to move the owner account to itself before creating a mailing list.
- RULES: the Vacation Rule priority is not set to 2, to make the server apply it before the Redirect All Rule.
- CLUSTER: the IMAP and ACAP backend login responses now carry the "Relay" flag.
- Bug Fix: 3.5: the Directory Integration option "generate mail attribute" did not work on non-Cluster systems.
- Bug Fix: WebUser: the Auto-wrap algorithm for "flowed" texts could cause crashes on some platforms.
- Bug Fix: HTTP: the Redirect operation placed "http://" into the Location header even if the current connection was an https one.
- Bug Fix: MacOS X (Darwin) package had a syntax bug in the "post-install" script.
- Bug Fix: Viewing the "Queue" Monitor page could cause Queue deadlock.

**3.5** 11-Dec-01

- POP: access to empty INBOX Mailboxes has been optimized.
- PIPE: Foreign Queue processing is implemented.
- PIPE: [STDERR], [FILE], [RCPT], and [RETPATH] tags are implemented.
- IMAP, POP: the STARTTLS/STLS option is seen in the CAPABILITY response only if the addressed Domain has the Security Certificate option enabled.
- Account Templates: the Initial Message text now can start with the [charset] prefix.
- Rules: now Reply and React texts can start with a [charset] prefix.
- WSSP: the ROUNDSIZE: prefix is implemented, the mailboxes.wssp page has been changed.
- WebUser (Skins): the directory processing has been changed, the directory.wssp and the sessiondirectory.wssp pages have been modified.
- Directory-based Domains: Skin support is implemented.

CLUSTER: header fields added with the frontend Server Rules are now stored by backends.
- Bug Fix: 3.5b9 incorrectly processed server-wide WebUser Preferences.
- Bug Fix: 3.3-3.5b9 a rare deadlock situation (in all prior versions) could stop ENQUEUER and DEQUEUER processors if an administrator opened the Message Monitor page.
- Bug Fix: 3.2-3.5b9 message file stored in the MDIR Mailboxes by Cluster backends could be improperly replaced with file links if the Reuse Temp Files option was enabled.
- Bug Fix: 3.5b5-b9: header composing algorithm for non-ASCII data could enter an infinite loop, blocking that thread and consuming CPU.
- Bug Fix: 3.4-3.5b9: supplementary send-phase relay checking algorithms were not the same as the input-phase algorithms, causing relay refusals in some rare situations.
- Bug Fix: 3.3-3.5b9: Directory: Local Unit: the delete record operation incorrectly checked if record children existed.
- Bug Fix: 3.4-3.5b9: CLUSTER: the POP3 "relaying enabled" flags were not processed correctly on frontends.

**3.5b9** 17-Nov-01

- LDAP/Admin: LDAP-based provisioning for regular domains is implemented (see the Directory Integration section of the manual).
- DNR: the "search PTR records" operation is implemented.
- SMTP: when message sending fails because the receiving host drops the connection, the message is re-enqueued (to avoid queue blocking for hosts that violate standards and just drop connections when they do not want to accept certain messages).
- SMTP: input messages exceeding the size limit are received, but they are not stored in files.
- Protection: the Banned Header and Body line settings are implemented. See the Protection section of the manual.
- Protection: the Client By Name option is implemented.
- Protection: the Unblacklistable (WhiteHole) Addresses list is implemented.
- Security: now Certificate Signing Requests can be generated and new Certificates can be set without prior removing of an existing certificate.
- CLUSTER: the cluster-wide Protection settings are implemented.
- CLI: Banned Header, Protection, and Cluster Protection commands are implemented.
- CLI: GETACCOUNTSUBSCRIPTION and SETACCOUNTSUBSCRIPTION commands are implemented.
- CLI: GETMAILBOXALIASES and SETMAILBOXALIASES commands are implemented.
- WebUser: more tags and tag parameters are cleaned out from HTML message portions now.
- WebAdmin: additional Monitor Access Rights are implemented.
- DNR: additional settings are implemented.
- DNR: requests to RBL servers are sent "quickly", so if an RBL server is down incoming SMTP connections do not time-out.
- WebUser: now when Sent and Drafts Mailboxes are auto-created, they are auto-subscribed to and the Mailbox list is refreshed.
- Bug Fix: 3.5b6-b8 versions crashed if an incorrect "LIST operation" address was used.
- Bug Fix: 3.5b6-8: WebSkins: password recovery E-mail address was not updated if password modification was disabled.
- Bug Fix: 3.5b6-8: WebSkins: custom Message header fields were not decoded from UTF-8 when a message page was being composed.
- Bug Fix: 3.5b8: External Helpers: crashed Helper could cause Server crash if the Helper log was enabled.
- Bug Fix: 3.5b7-8: LOCAL Delivery: some error codes could be lost, so no error reports were sent back when delivery to an account failed.

**3.5b8** 03-Nov-01

- SMTP: the Advertise NTLM AUTH option is implemented.
- SMTP: now when the module sends the STARTTLS command, it uses the SSLv3 (rather than SSLv2) "hello" operation.
- DEQUEUER: more SNMP statistics elements have been implemented.

- DEQUEUER: minor internal algorithm changes.
- DIRECTORY INTEGRATION: Server-Wide and Cluster-Wide settings are implemented.
- CLI: the [GET|SET][CLUSTER]DIRECTORYINTEGRATION commands are implemented.
- CLI: the Skin Administration commands are implemented.
- CLI: the GETCLIENTIPS and GETBLACKLISTEDIPS commands are implemented.
- CLI: the GETWEBUSERSESSION and KILLWEBUSERSESSION commands are implemented.
- LOCAL: if a message is delayed by a Cluster backend, the entire account queue is suspended now.
- LOCAL: the SNMP statistics elements have been implemented.
- CLUSTER: slave startup procedure has been changed to avoid problems on systems with a large number of shared domains.
- SNMP: processing of "not-found" elements have been changed to match the SNMPv2 specs.
- HTTP: the "SkinFiles" realm is implemented to allow Skin File retrieval without using a WebSession URL.
- WebUser: auto-wrap algorithms have been changed.
- WebUser: the autoWrap "flowed" option has been added.
- WebUser: the charset parameter of message parts containing the ASCII-only symbols is ignored now.
- Queue: the Web Monitor modules have been changed to display not more than 1000 items in huge queues.
- IMAP: non-standard parameters of the Content-Type and Content-Disposition fields are retrieved now.
- Foundation: thread priority routines are implemented.
- Foundation: Linux: the STTask routines have been changed to avoid leaving "zombies" of killed processes.
- Foundation: BSD: the STDirectoryEnumerator routines have been changed to avoid directory-detection problems on NFS filesystems.
- Bug Fix: Foundation: in 3.5b7 the setInetAddress routine incorrectly formatted IP addresses as numeric strings. This could cause problems in specifying IP addresses on the Listener, Cluster, and SNMP WebAdmin pages.
- Bug Fix: SNMP: the value of parameters that changed their types in 3.5b7 from INTEGER to COUNTER was sent incorrectly via SNMP.

**3.5b7** 24-Oct-01

- SNMP: the WebAdmin Monitor interface to SNMP data is implemented.
- SNMP: the "total number of jobs"-type parameters now have the Counter data type.
- SNMP: HTTP monitoring elements have been added.
- SMTP: the Force AUTH option is implemented.
- WebAdmin: Admin account Preferences processor has been changed.
- Directory: the "search"-type operations now enter the subtrees stored on different Units.
- CLUSTER: the Cluster-Wide Rules are implemented.
- CLI: the GETCLUSTERRULES and SETCLUSTERRULES commands are implemented.
- CLUSTER: the Cluster-Wide Router Table is implemented.
- CLI: the GETCLUSTERROUTERTABLE and SETCLUSTERROUTERTABLE commands are implemented.
- CLI: the GETACCOUNTEFFECTIVESETTINGS and GETDOMAINEFFECTIVESETTINGS commands are implemented.
- CLI: the [GET|UPDATE|SET][ACCOUNT|DOMAIN] commands have been renamed into the [GET|UPDATE|SET][ACCOUNT|DOMAIN]SETTINGS commands. Old names continue to work, too.
- WSSP: the EQUALS operation with a quoted-string argument is implemented.
- Migration: the MoveAccounts program and its parameters have been changed.
- LDAP: the modifyDN "newRDN" parameter was processed incorrectly.
- AIX: build parameters have been modified to support 6000+ threads.
- Bug Fix: WebUser: the number of selected messages for the mailbox.wssp page was calculated incorrectly.
- Bug Fix: Directory: Remote Units: the Search operation returned "unstripped" DNs when the "Server Base" setting was non-empty.

**3.5b6** 18-Oct-01

- Security: the login-disabling options protecting Accounts from Password Attacks are implemented.
- RPOP: when retrieving mail from Unified Domain-Wide Accounts without using Special Headers, the module

now checks that To:/Cc: addresses can be routed to a Local account (rather than just checking that they are directly addressing the Main Domain).

- CLUSTER: Cluster-wide Default Domain Settings, Default Account Settings, Alerts, and WebSkins are implemented.
- CLI: the Cluster-wide versions of the commands dealing with Default Domain and Default Account settings are implemented.
- CLI: the Alert Administration commands are implemented.
- WebUser: URLs for non-ASCII attachments are now composed in the UTF8 charset (to work with Windows Internet Explorer).
- WebUser(WebSkins): the Address Books records are sorted now.
- WSSP: the RANDOMELEMENT function is implemented.
- Domains,WebUser Settings: non-ASCII attributes are supported now.
- MoveIMAPMail: the --target option has been implemented.
- Bug Fix: new (Skin-based) List Archive Browser pages did not have the correct charset specified.
- Bug Fix: old (Web-User and WebAdmin) List Subscribers pages could crash the server if the list had the Require Confirmation option disabled.
- Bug Fix: Skins: the default rules.wssp and rule.wssp files did not have the closing </SELECT> tag, causing problems for Netscape browsers.
- Bug Fix: Skins: the webUserSiteIndex internal code routine misplaced the account name (was shown as an error code).

**3.5b5** 07-Oct-01

- SMTP: the Wake up Now button has been added to the SMTP Settings WebAdmin page.
- DNR: the new Custom setting allows an administrator to specify DNS Server addresses explicitly. See the SysAdmin section for the details.
- WebAdmin: Preferences now work for Secondary Domain Administrators, too.
- WebAdmin: the Charset parameter has been added to the Administrator Preferences.
- WebAdmin: Alerts now stored using the UTF-8 charset.
- Directory: spaces around the comma signs are removed from the DN strings.
- LIST: when archive Mailboxes are swapped, the newly created archive Mailbox gets the ACLs of the old archive Mailbox.
- Mailboxes: the "Redirected" message status is implemented.
- Groups: non-ASCII "real names" are supported now.
- Accounts: non-ASCII "real names" and "custom attributes" are supported now.
- Account Templates: non-ASCII "real names", "custom attributes", and Mailbox names are supported now.
- WebUser: the "Redirected" message status is set when a message is redirected or forwarded.
- WebUser: renaming and remove Mailboxes when the Show Subscribed option is enabled now renames/removes the Mailbox and submailboxes from the Subscription list.
- WebUser: now the Composer encodes non-ASCII attachment file names.
- Bug Fix: the Save Sent Messages Mailbox could not be set in the 3.5b4 Skins interface.
- Bug Fix: the 3.5b4 version incorrectly processed most non-cluster License Master Keys.
- Bug Fix: Rules: if the Reply/Reply All operation parameter was specified with the "+" sign and additional headers, the Cc: headers were not processed at all, and Bcc: headers caused parsing errors.
- Bug Fix: Mailbox renaming for non-top level Mailboxes did not work with the "rename Submailboxes" option.

**3.5b4** 04-Oct-01

- WebSkins are implemented. Now all domains that do not have files in the Account subfolder of the old custom WebUser Interface directory use the new Skins Interface.
- LIST: additional subscriber address checks are implemented to avoid self-subscribing of the special mailing list addresses (-on, -subscribe, -off, etc.)
- CLI: the SetAccountPassword command is implemented.
- SMTP: the "send encrypted wherever possible" option is implemented.

- RBL: RBL responses in the 127.1.x.x range are now recognized as "blacklist it" responses.
- Migration: the utility to simplify migration from the Post.Office product has been implemented.
- WebAdmin: the `/MainAdmin/` realm is implemented (see the HTTP section of the manual).
- Directory: the CAChain attribute has been added to the default Schema.
- LDAP: the Compare operation is implemented.
- ROUTER: the parser has been changed to allow special symbols in the left part of the alias records (records like `<FAX=*> = *` can be used now).
- ROUTER: now account alias records can be used for the Central Directory-based Routing.
- Bug Fix: WebUser/WebAdmin: in 3.4b3 version custom file uploading could fail in regular (non-directory-based) Domains.
- Bug Fix: In 3.5b2-3 the "Insert All" Directory Integration operation could crash the server if the Domain had at least one Group.
- Bug Fix: CLUSTER: renaming a shared domain did not rename Domain Aliases on all cluster members.

**3.5b3** 25-Jul-01

- Many internal file-handling routines have been changed.
- WebAdmin: Log settings can now be changed only if the Administrator has the CanModifySettings Server Access rights.
- RPOP: now the Leave On Server option works with any types of UIDs remote servers present and the list of retrieved UIDs is preserved between Server restarts.
- RPOP: the APOP option is added to individual RPOP records.
- Local: the `X-Special-Delivery: test` header field now works with the Local Delivery module, too: messages with that field are not stored in the Account Mailboxes.
- Forwarders and Groups: processing algorithms have been changed.
- Directory-Based Domains: Groups are supported now.
- Directory: the mailListSettings and groupOptions attributes have been added to the default Directory Schema.
- Directory-Based Domains: Custom WebUser and WebAdmin Interfaces are supported now.
- Directory: the fileData attribute and the CommuniGateWebInterface objectClass have been added to the default Directory Schema.
- Directory: the mailListSettings and groupOptions attributes have been added to the Directory Schema.
- SMTP: a workaround for sending mail to buggy firewall relays is implemented.
- List: the "Hide From Address" option is implemented.
- PWD: when a user with the Server Settings access right logs in, the max input buffer size is increased to 1MB (to allow for larger data in the Router operations).
- Bug Fix: WebUser: non-standard MIME content types and subtypes could be returned corrupted.

**3.5b2** 15-Jul-01

- DEQUEUER: algorithms have been changed to avoid crashes when multiple DEQUEUER threads process the same message.
- IMAP: the FETCH algorithms have been changed to avoid loading large messages into memory.
- CLUSTER: the "virtual Mailbox" code has been redesigned.
- SECURITY: the SASL NTLM method now works with Macintosh versions of Microsoft products (Outlook Express, Entourage).
- WebSite: nested folders are supported now. The WebAdmin WebSite.html and WebUser WebSite.html and UserSiteIndex.html files have been updated.
- ADMIN: now Domain Administrators can control several Domains. See the SysAdmin section of the manual for more Details.
- CLUSTER: now Mailing List Archives can be browsed via any cluster member.
- CLUSTER: now the GETLIST and UPDATELIST commands work with any Cluster member.
- CLUSTER: now the Create/Rename/remove operations for Forwarders and Groups work with any Cluster member.
- STATIC CLUSTER: Accounts in Shared Directory-based Domains can be administered from any Cluster

member.

- WebUser: attachments in the AppleDouble format are displayed correctly.
- SMTP: the TURN command is implemented to support dial-up client sites running Microsoft Exchange servers.
- DIRECTORY: the folderIndex and adminDomainName attributes have been added to the CommuniGateDirectoryDomain objectClass in the default Schema.
- Bug Fix: CLUSTER: WebAdmin: Personal WebSite administration did not work across Cluster members.
- Bug Fix: ROUTER/DNR changes in 3.5b1 effectively disabled the SMTP RBL feature.
- Bug Fix: RULES: the Reply/React operations in 3.5b1 incorrectly inserted the From:/Sender: header field.

**3.5b1** 06-Jul-01

- Security: the SASL NTLM authentication method is supported now (this method allows you to use the "Secure Password Authentication" option in Microsoft products).
- POP: the parameterless AUTH command is supported now (for MS Outlook compatibility).
- Startup: Unix startup scripts have been modified to support a custom Startup.sh file in the {Base} directory.
- SMTP: now the ATRN command is always accepted from non-client addresses (after the AUTH command).
- Initialization: now garbage collector is activated during domain initialization, so systems with several thousand domains and 10,000+ domain aliases do not abuse VRAM on restart.
- SMTP, RPOP, LDAP, Router: all settings that should contain an A-record domain name (such as the forwarding server name) can contain several explicitly specified IP addresses, separated with the comma sign.
- LDAP: Start TLS (RFC2830) is supported now.
- LIST: RFC2919 (LIST-ID header field) is implemented.
- Accounts: the Date: header field is now added to the Initial Message when it is stored in the INBOX of newly created accounts.
- CLI: the LISTSUBSCRIBERS command is implemented.
- CLI: the WRITELOG command is implemented.
- WebUser: the "Trash" Mailbox can now be replaced with a Mailbox alias (useful to create a "shared Trash" Mailbox for several accounts).
- Bug Fix: Cluster: the "addMessages" operation did not work correctly with Virtual Mailboxes.

---

**3.4.8** 26-Jun-01

- Bug Fix: Cluster: certain Rule operations could cause the Local delivery protocol de-synching and backend crashes.

**3.4.7** 23-May-01

- LOCAL: delivery is now repeated in 1 minute if an External INBOX is locked with some other application.
- CLUSTER: backend Local Delivery now checks the Mail Disabled and Account is Full conditions.
- WebUser: the "Send as HTML" option is added to the Compose.html page.
- WebUser: the converter now displays embedded objects in HTML messages [incorrectly] created with Lotus Notes.
- WebUser: now the Drafts Mailbox is auto-created during the "Save as Draft" operation.
- SMTP: the "STARTTLS" EHLO response is now presented only if the target domain has an active Certificate.
- Bug Fix: WebUser: updating Mailbox ACLs from a non-owner account could clear the ACL list.
- Bug Fix: The process environment variables were not correctly passed to external programs.
- Bug Fix: SMTP: the ATRN command was issued instead of ETRN if the ATRN 'loginname' was entered into the module settings.
- Bug Fix: WebUser: large Subject lines could result in insertion of a header string with just one space symbol.
- Bug Fix: WebUser: incorrect customization of the Compose page could cause server crashes.

**3.4.6** 05-May-01

- Windows NT/2000: OS User Names that contain the '%' sign can now be used to explicitly specify the Windows

Domain that should be used for Authentication.

- Directory: Local Units can now insert records that have multiple objectClasses specified (if one specified class is a child of all other specified classes).
- Directory: Local Units now ignore spaces in the first parts of the DN elements.
- Bug Fix: CLUSTER: The Incoming flow control operation could cause a crash of a backend server.
- Bug Fix: IMAP: the FETCH BODY[] operation could return message text without headers if the same FETCH operation included the BODYSTRUCTURE keyword.
- Bug Fix: SMTP (3.4.5) failure to create an output stream could destroy the SMTP module queue and crash the server.
- Bug Fix: RULES: the Each/Any ROUTE conditions were not available on all OS platforms (compiler-related error).

**3.4.5** 22-Apr-01

- Windows NT/2000: CGStarter application now can accept and remember parameters set in the Services control panel.
- SMTP: now secure connections can be used to connect to backup (ETRN/ATRN) servers.
- SMTP: now the HELO and Return-Path parameters are not verified for connections coming from Dynamic Cluster members.
- Foundation: On MS Windows platforms NICs with more than 30 network aliases (IP addresses) are supported now.
- Bug Fix: SMTP: Send Encrypted option did not work for the messages that had to be sent via a specific Local IP address.
- Bug Fix: SMTP: the Send Encrypted feature did not re-send the EHLO command causing problems for some freeware MTAs.
- Bug Fix: SMTP: the DSN parameters were incorrectly used when sending messages with empty return paths.
- Bug Fix: WebAdmin: the Account Import function did not check the Domain Administrator access rights.
- Bug Fix: External Filtering caused a memory leak.

**3.4.4** 09-Apr-01

- LOCAL: the Flow Control settings are implemented.
- Listener: the Maximum Connections from the Same Address setting is implemented.
- SMTP: the module can now send the AUTH command to the Forwarding Server.
- Rules: messages generated with the Reply and React operations now use <Mailer-Daemon@maindomain> return-path address.
- SNMP: local IP address selection and remote IP address restriction settings are implemented.
- The --SharedFiles command line option is now supported for MS Windows platforms.
- AUTH: the DIGEST-MD5 SASL method is correctly re-implemented now.
- CLI: the SetClientIPs and SetBlacklistedIPs commands are implemented.
- SMTP: messages with extremely long lines (more than 100K) do not result in connection terminations now.
- Logs: The Auto-delete setting can be set to 1 year.
- Rules: now the [RETPATH] prefix can be used with the Execute actions.
- External Filtering: License Key management mechanism has been modified to avoid processing delays after server restart. A new version of McAfee Plugin is required to benefit from this modification.
- WebUser: the content of "message/delivery-status" MIME part is decoded and displayed now.
- WebUser: the Compose function now tries to break long To/Cc/Bcc/Subject header lines into multiple shorter MIME lines.
- Bug Fix: Account: attempts to create foreign Mailboxes with empty names could crash the server.
- Bug Fix: SMTP: the AUTH command sent before the ATRN command could incorrectly form the password string.
- Bug Fix: Domains could not be renamed ("domain is in use" error) if they had at least one mailing list.

**3.4.3** 25-Mar-01

- SMTP: the release queue methods (ETRN, wakeup-email) now release both the generic domain queues and the 'send-via-this-ip' domain queues.
- Directory-based Domains: the WebAdmin interface now implements the Load New Domains operation.
- CLI: the CREATEDIRECTORYDOMAIN and RELOADDIRECTORYDOMAINS commands are implemented.
- Foundation: support for both process-level and system-level thread scheduling is implemented.
- ADMIN: the --ThreadScope command line parameter has been added.
- RULES: Reply With operation not adds a space to the "Re:" subject prefix.
- ACCOUNT: FreeBSD-style MD5-encrypted passwords are supported now.
- CLUSTER: time-out values in Dynamic Cluster operations have been modified to ensure proper failover when the Controller network connection fails.
- WebUser: the Forgotten Password page now passes the current Domain Name to Cluster Backends.
- WebUser: the Mailbox.html page now supports the NextMessage=msgid and PrevMessage=msgid parameter to provide the "next/prev" messages switches. The NextPrevJump.html file is added.
- WebUser: processing of multipart/alternative messages has been improved (added support for Apple's Mail.app message formatting).
- Bug Fix: Mailboxes: the LIST command incorrectly checked the access rights for nested submailboxes in foreign accounts (some foreign Mailboxes could be invisible even if the user had the Lookup right for them).
- Bug Fix: The MoveIMAPMail utility did not copy empty Mailboxes from some 3rd party IMAP servers.
- Bug Fix: LIST: when the number of LIST processors was decreased using CLI, the excessive processors did not disappear.
- Bug Fix: WebAdmin: updating the Obscure page settings could change the Helpers page settings.
- Bug Fix: Directory Based Domains: if the automatic account creation procedure failed, the cleanup algorithm could end up in a deadlock.
- Bug Fix: WebUser: the Login page did not show the "forgotten password" link is the anti-harvesting option was enabled.

**3.4.2** 06-Mar-01

- U-crpt password encoding now works on AS/400 and BeOS platforms.
- The Dynamic Cluster Monitor page is added to the WebAdmin Interface.
- CLUSTER: the Dynamic Cluster Controller fail-over algorithm is improved.
- Bug Fix: the 3.4.1 version did not properly accept the License Keys.
- Bug Fix: CLUSTER: attempts to rename or remove an account in a shared Domain after a Controller fail-over could crash the new Controller.
- Bug Fix: CLI: the RenameDomain command did not work properly in 3.3/3.4b versions.
- Bug Fix: KERNEL: delivery to groups improperly decreased the Domain usage counters. This bug produced the "open counter < 0" messages in the Server and system logs.
- Bug Fix: LIST: the text/alternative Posting Format restriction option incorrectly checked the message structure.

**3.4.1** 01-Mar-01

- Group and Forwarder creating/renaming algorithms have been changed.
- Shared Domains: special mailing list addresses (-report, -on, -off, etc.) now work in Shared Domains.
- TLS: max output block size is decreased to provide a workaround for products using Microsoft SSL libraries.
- LIST: the HELP command is implemented.
- IMAP, MIME: the Content-Description field is processed now.
- WebUser: the Reply operation now uses a smarter algorithm to find plain text message portions.
- WebUser: the Strings.data file now contains the "Translator" dictionary that can be used to translate pop-up menus like those used in the Rule composer.
- WebUser: the ^B and ^C macros are removed from the Rules.html and Rule.html pages (the buttons can be customized now).
- HTTP: the Personal Web Site prefix is now detected after URL decoding.
- Mailboxes: now all 'outer' Mailboxes are automatically created when a sub-mailbox is created.
- Bug Fix: WebUser: the default reply text routine could incorrectly insert quotation marks. In some rare cases it

could cause a system crash.

- Bug Fix: On Unix platforms the external programs did not get all environment parameters.
- Bug Fix: some Server Settings updates could cause false 'main domain renaming' operations.
- Bug Fix: 'mailbox list' routine incorrectly capitalized names of 'xxxx/inbox' sub-mailboxes.
- Bug Fix: Log recording stopped if the total size of all logs generated since the last server restart exceeded 4GB.
- Bug Fix: Domains could not be removed if the domain Default WebUser Preference set has been used at least once.
- Bug Fix: Server-Wide Rules updates could crash the server when a very long rule operation (as a suspended virus scan) was in progress.

**3.4** 18-Feb-01

- Directory Integration: the `mail` attribute processing can be fine-tuned now. See the LDAP Module chapter for the details.
- CLI: the StatReset keys were added to the Domain and Account Statistics data.
- CLI: the CREATEWEBUSERSESSION command is implemented.
- Forwarders: addresses without '@' and '%' symbols are qualified using the forwarder Domain name.
- WebUser: the "Print-friendly" message link is implemented, the Strings.data "MessageHeaderEnd" string has been modified.
- Bug Fix: MAILBOXes: in 3.4b9 'rollbacking' the target Mailbox after failed copying operation could crash the server.
- Bug Fix: WebUser: not all special symbols in Mailbox names were properly URL-encoded.

**3.4b9** 04-Feb-01

- Directory Integration: the 'Store Passwords for Regular Accounts' option is implemented.
- LIST: the Content-Transfer-encoding header field is removed from distributed 'feed' messages only if its value is '7bit' (previous versions removed the header with the value of '8-bit', too).
- IMAP: the MULTIAPPEND extension is implemented.
- CLUSTER: now errors on inter-server SMTP connections do not turn on delays and other anti-attack mechanisms.
- Passwords: the --BatchLogon command line parameter is documented.
- Passwords: now the U-crpt password encryption can be used on Windows-based servers, too.
- Bug Fix: Domains: when the Main Domain was renamed, the Domain Directory Subtree was not renamed.
- Bug Fix: WebAdmin: the size-type parameters could not be set to 'Default (XXX)' values.
- Bug Fix: Messages with several 'Subject:' headers were processed incorrectly.
- Bug Fix: LIST: the 'banned' mode could not be set for lists with disabled archiving.
- Bug Fix: WebUser: if a custom WebUser Interface file was empty, the server could crash.

**3.4b8** 06-Jan-01

- Directory Integration: the "UID Subtree" setting is implemented.
- SMTP: Relay To Client settings can be set to "no".
- Domains/External AUTH: the Consult External Authenticator Domain Settings is implemented.
- Admin: the "Drop Server Root privilege" options are implemented (Unix only). See the SysAdmin section for the details.
- DEQUEUER: dequeuer messages can be customized now (using the General->Strings page).
- POP: the message size has been added to the RETR command response (to make Netscape mailer show its progress bar correctly).
- Account: the Collect Account Statistics setting is implemented (the Obscure page).
- Domain: statistics on received messages is collected now.
- CLI: GETACCOUNTSTAT, RESETACCOUNTSTAT, GETDOMAINSTAT, and RESETDOMAINSTAT commands are implemented.
- MAILBOX: the "~username" Mailbox aliases are implemented now - they provide access to all shared

Mailboxes in the specified account.
- Foundation: the OS Password checking routines now check for the OS Account and Password expiration dates.
- External AUTH: Logging changed.
- Bug Fix: INFOWEBFILES renamed into GETWEBFILESINFO, output format of GETWEBFILE and LISTWEBFILES was incorrect.

**3.4b7** 03-Dec-00

- Mailbox: the Mailbox view creation algorithm has been changed.
- Directory: Remote Units: the Server Subtree setting is implemented.
- Directory: many attributes and objectClasses have been added to the Local Unit built-in Schema.
- Directory: the Import LDIF and Import LMOD functions are implemented in the WebAdmin Interface.
- WebUser: the X-UUEncode MIME encoding is supported now.
- MIME: UUencoded files embedded into plain text messages are recognized now.
- CLI: GETWEBFILE, LISTWEBFILES, INFOWEBFILES, PUTWEBFILE, DELETEWEBFILE, and RENAMEWEBFILE commands are implemented.
- WebUser: the UUencoded files embedded into plain text messages are displayed now.
- Account: the new method is used to recalculate the MailStore size stored in the .info files.
- Account Admin: now Account Rules can be specified in the Account Import file.
- Directory-based Domains: Personal WebSite support is implemented.
- Directory-based Domains: Account Removal is implemented.
- Directory-based Domains: Account Renaming now renames the Account files, too - if they reside within the DirectoryDomains file directory.
- Directory-based Domains: Forwarders are implemented.
- Template: Initial (default.html) page for Personal WebSites can be specified now.
- WebUser: the keywords INBOX and Trash can be "translated" in the Strings.data file now, so these "fixed-name" Mailboxes can be "renamed on screen".
- WebUser: the HeaderNames dictionary is added to the Strings.data file. It can be used to change the names of the RFC822 header fields used on the Mailbox and Message pages.
- Router: local routing algorithms have been changed to restore compatibility with the old router.
- POP: the login response line contains the total size of all messages in the Mailbox (to make the Netscape e-mail client happy).
- CLUSTER: Personal Web Site support now works in both Static and Dynamic Clusters w/o exceptions.
- CLUSTER: Domain Aliases are now properly initiated during the frontend and "slave" backend startups, and the domain aliases are properly removed from all servers when the domain is removed.
- CLUSTER: non-local IP Addresses can be assigned to shared domains in Static Clusters.
- Bug Fix: TLS: the 'exportable' TLS 1.0 methods were implemented incorrectly in 3.4b6.
- Bug Fix: WebAdmin/WebUser: the default numeric settings could be displayed as "-2" strings.

**3.4b6** 24-Nov-00

- Directory: the Search Results Limit setting is added to the Local Unit settings.
- WebUser: the Session Time Limit setting is implemented.
- LIST: the "Special" posting mode processing has been changed to make it possible to subscribe lists to other lists.
- Rules: the [RCPT] Execute operation prefix is implemented.
- RPOP: processing is re-scheduled when an RPOP record changes its poll period value.
- Directory-based Domains: when account is auto-created on first access, the INBOX Mailbox and other account details (suppl. Mailboxes, Mailbox aliases, subscriptions, etc.) are automatically created, too.
- Template: Initial (Greeting) message can be specified now.
- CLI: GETACCOUNTLISTS and GETDOMAINLISTS return value types are changed to *dictionary*.
- TLS: TLS 3.1 is implemented.
- Queue: delivery delay warnings are implemented.
- SMTP: relaying settings have been modified.

- SMTP: the Send Warnings setting is implemented.
- Local Delivery: the Send Warnings setting is implemented.
- Bug Fix: 3.4b4-b5 versions did not store the updated Queue page Settings on disk.
- Bug Fix: 3.4b versions could show duplicate IP Addresses assigned to Domains.
- Bug Fix: TLS sessions interrupted in the negotiation phase could crash the server.
- Bug Fix: in 3.4b5, mail sent to an incorrect mailing list address could crash the server.

**3.4b5** 18-Nov-00

- Domains: the global DomainAliases.tdb file is phased out. Now the DomainAliases.data file in the Settings directory of the Domain subdirectories is used to store this domain aliases. The content of DomainAliases.tdb is moved automatically to those files, and the DomainAliases.tdb file is renamed into DomainAliases.tdb.unused.
- Dynamic Cluster: the Domain Aliases now work for Shared Domains, too.
- Kernel: the old DataBase Managers (handling the .tdb and .ldb files) are phased out.
- ROUTER: algorithms used for Local Domain Routing are redesigned.
  **Note**: this version can act as a Cluster Controller for 3.4b3-3.4b4, but not vice versa. Complete Cluster Upgrade is required.
- Static Cluster: Directory-based Domains can be used for Static Clustering now.
- Domains: the Send To Forwarders option has been added to the Mail to All panel.
- WebAdmin: Administrator Preferences are implemented (see the SysAdmin->HTTP section of the manual for more details).
- WebAdmin: the Preferences settings are implemented for POP,IMAP,ACAP,LIST,SMTP,LOCAL,PIPE, and RPOP Monitors.
- WebAdmin: the Preferences settings are implemented for the Domain Account List page.
- WebAdmin: all numeric settings can now be set to some "other" (unlisted) value.
- IMAP: RFC2971 (the "ID" extension) is implemented.
- LISTS: the GETACCOUNTLISTS command is documented.
- CLI: the GETDOMAINLISTS is now the preferred name for the LISTLISTS command. This command now works in the Dynamic Cluster environment.
- CLI: the MAINDOMAINNAME command is implemented.
- LIST: delivery to Lists now works without exceptions in both Dynamic and Static Clusters.
- LOCAL: delivery to "all" now works without exceptions in both Dynamic and Static Clusters.
- PIPE: the Queue (Wait) page is added to the WebAdmin Monitors.
- Bug Fix: the default Directory Schema did not contain the "cn" attribute for the "organization" objectClass. In 3.4b4 version, this bug made domain Directory record creation impossible. On fresh installations, the "postmaster" account was not created.

**3.4b4** 09-Nov-00

- Domains/SMTP: the Local IP Address for outgoing SMTP connections can be specified now. See the SMTP and Domain Settings sections for the details.
- CLI: the DELETEMAILBOX, RENAMEMAILBOX commands are implemented.
- IMAP, WebMail: foreign Mailboxes can be renamed and deleted now.
- QUEUE: the Copy Failure Reports option is implemented.
- ACAP: the PREFIX, SUBSTRING, and SUFFIX filters are implemented.
- SMTP: the server now reports the "DNS Loop" situation only if a remote host name resolves into an IP Address the SMTP Listener is enabled for.
- SMTP: the domain name used in the ATRN command must be included into the `Hold Mail for Domains` list.
- Central Directory: the hostServer attribute can be renamed now.
- CLI: the RefreshOSData, GetRouterTable, and SetRouterTable commands are implemented.
- Rules: The Redirect All simplified Rule now has the "Preserve To/Cc fields" option.
- Central Directory: forwarder records are now included into the Domain directory subtree.
- Directory: Local: the Enforce Schema setting is implemented (it is now enabled by default).
- WebUser: if the Save a Copy option is selected on the Compose page and the Mailbox with the specified name

does not exist, a new Mailbox is automatically created.

- Streams: the PLAIN authentication method now uses an empty challenge string.
- Migration: the MoveIMAPMail and MovePOPMail utilities now support the --noTimeOut flag.
- Bug Fix: ACAP/IMAP: if the AUTHENTICATE command was interrupted, the NO response was returned instead of the BAD response.
- Bug Fix: Directory: Local Units incorrectly processed some update requests ("add" attribute sets were processed as "replace" sets).
- Bug Fix: OS Passwords did not work on the AS/400 platform.

**3.4b3** 30-Oct-00

- CLUSTER: a separate Domain Controller is implemented.
- CLUSTER: the Account Controller is completely redesigned.
- CLUSTER: the Account and Domain Controllers now automatically move to a different backend server if the current Controller backend server fails.
- CLUSTER: error codes reported by backends are now transparently relayed via frontends to client mailers.
- CLUSTER: POP Alert messages generated on backends are now relayed via frontends to client mailers.
- CLUSTER: backend IMAP alerts issued at the login time are now passed to the client.
- DIRECTORY: the "cn=schema" subtree is implemented. The Local Unit schema can be retrieved and extended using the 'cn=schema' record.
- Foundation: the STSkipList data structure is implemented.
- Several internal routines switched to the STSkipList structures to improve performance on large systems (1,000,000+ accounts, 10,000+ domains in queue, etc).
- SECURITY: the Hide Unknown Account Error option is implemented and it is enabled by default. See the Security section for more details.
- SMTP: the Advertise 8BITMIME option is implemented.
- Bug Fix: the server could crash if someone modified the Domain Settings while a domain Private Key was being generated.
- Bug Fix: ACAP: data strings with special symbols are returned as literals now.

**3.4b2** 06-Oct-00

- EXTFILTERING: McAfee scanning plugin is implemented.
- SNMP: the CGatePro-MIB.txt file now includes the absolute OIDs.
- WebUser: The Thai charsets support is implemented.
- TLS: the Generate Key option is added to the Domain Security page.
- TLS: support for 2048-bit keys and long Certificates is added.
- TLS: the Certificate Authority Chain option is implemented.
- TLS: the nested TLS negotiations are supported now. They are used to implement strong (128-bit) SSL encryption with weak (40-bit), "export-legal" products.
- Shared Domains: if LDAP connections to the Central Directory fail, the SMTP module now returns a non-fatal error code for unroutable addresses.
- Bug Fix: Directory: Multi-Level searches in Local Units could return incomplete DNs. This could also cause the LDAP module to stop on-the-fly creation of the "mail" attributes.
- Bug Fix: incorrect parallel initialization could cause the PIPE module to crash on startup.
- Bug Fix: the RPOP module could accept To: and Cc: addresses as "trusted" when working without the "Special-header" option (this could result in unwanted relays).
- Bug Fix: ENQUEUER: the default value for ENQUEUER threads is 1 now.

**3.4b1** 07-Aug-00

- QUEUE: ENQUEUER design is multi-threaded now.
- HELPERS: Content Filtering/Anti-Virus API is implemented (see the Rules section).
- CLUSTER: "Virtual" Mailbox objects are implemented (allowing an account opened on one server to access

- Mailboxes in an account opened on a different server).
- CLI: LISTMAILBOXES, CREATEMAILBOX, GETMAILBOXINFO, GETMAILBOXACL, SETMAILBOXACL, and GETMAILBOXRIGHTS commands are implemented.
- Mailboxes: ACL subsystem has been redesigned.
- Mailboxes: MDIR Mailboxes are now parsed correctly even if the message internal date was set to 0.
- CLUSTER: PWD module now returns different codes for some key error messages.
- Accounts: excessive file operations are removed from the account opening procedure.
- WebUser: Japanese (ISO-2022-JP) characters are now correctly processed in the message header fields on the Compose page.
- WebUser: The UTF8 mode for the Japanese (Big5 and GB2312) encodings is supported now.
- WebUser: The UTF8 mode for the Japanese (ISO-2022-JP) encodings is supported now.
- Notifier: the Log Level and Queue size can now be specified using the Obscure page.
- Security: External Authenticator internals are redesigned, its settings are moved to the Helpers page.
- Bug Fix: Personal WebSite: URLs for site files did not contain URL escape symbols, some of the access utilities did not remove the URL escape symbols.
- Bug Fix: POP, IMAP: SASL AUTH methods incorrectly supported the "short-form" syntax.
- Bug Fix: CLUSTER POP login could fail on backends.

---

### 3.3.2 02-Oct-00

- Security: the Certificate processing buffer size has been increased from 1K to 4K.
- Bug Fix: SMTP: Dynamic Cluster backends could crash when the Mailbox STORE operation failed.
- Bug Fix: incorrect parallel initialization could cause the PIPE module crash during startup.

### 3.3.1 07-Aug-00

- BeOS version is released.
- Bug Fix: WebUser: processing format=flowed texts could cause crashes on some platforms (AS/400).
- Bug Fix: SMTP: malformed ETRN could cause crashes.
- Bug Fix: CLUSTER SLAVE: the controller response parser could crash the server.
- Bug Fix: LIST: automatically-generated messages were processed incorrectly.
- Bug Fix: UTF8/Unicode decoding procedure did not work correctly for several charsets.

### 3.3 16-Jul-00
**Update Note**: the 3.3 version uses a completely new Directory Manager. If some of your Domains had the Directory Integration setting set to Keep In Sync, open those Domain Settings in the newly installed 3.3 version, and click the "Insert All" button in the Directory Integration Panel.

- Domains: Mailing lists and Groups are now automatically added/updated in the Directory if the Domain Integration setting is set to Keep In Sync.
- Domains/Accounts: now OS Names can be explicitly set for individual Accounts.
- HTTP: CGI programs now inherit the environment variables of the Server (under Windows, this is needed to open TCP/IP sockets in CGIs).
- Rules: the "[FILE]" and "[STDERR]" Execute command tags are implemented.
- IMAP: the APPEND command now checks if the message text lacks the trailing EOL and fixes it. This is a workaround for the Netscape Messenger bug.
- Groups: the Remove Author Address option is implemented.
- Groups: sending to a non-empty group with all group addresses removed is processed as normal (final) delivery now.
- ACAP: datasets entry names are case-insensitive now.
- WebUser: the Sent and Draft Mailbox names are properly "defaulted" now. The WebUser Settings.html page has been changed.
- WebUser: the simplified Chinese (GB2312) charset is supported now.

SNMP: 64-bit Counters are implemented, some MIB elements have been switched to the COUNTER format.

- Bug Fix: Directory: Browser: URL escape symbols were not removed correctly, causing problems for Netscape browsers.
- Bug Fix: POP: the CAPA command was not processed in the TRANSACTION state.
- Bug Fix: binary zeros in message headers could crash the server.

**3.3b9** 06-Jul-00

- Manual: the Directory, Directory Integration and Clusters pages are updated.
- WebAdmin: Obscure: the Central Directory settings are phased out.
- Directory: File(Local Units): the updates merging daemon is implemented.
- Directory: Storage Unit deletion is implemented.
- Directory Integration: the Delete All operation now removes only the records for accounts created on this Server.
- WebUser: Korean (ISO-2022-KR) and Chinese (Big5) letters are correctly converted into HTML code.
- WebUser: the Certificate link is displayed only if the domain has a Custom Certificate.
- WebUser: the Mailing Lists link is displayed only if the domain has some mailing lists.
- Foundation: a faster version of 'write to file' method is implemented (now used to store aliases, forwarders, groups, and account info).
- LDAP: case-insensitive dictionaries are now used in modify-type operations.
- HTTP: CGI environment variables HTTP_AUTHORIZATION and HTTP_REFERER are added.
- HTTP: CGI program name in a URL can now be followed by '/' and some URL string.
- Bug Fix: Mailboxes: 3.3b6-b8 versions might not show INBOX in the Mailbox list, if INBOX was an external Mailbox.
- Bug Fix: Routing: Directory-based Routing did not work in 3.3b3-b8.
- Bug Fix: under IRIX, AIX, HP/UX daylight saving times for the local time zones was not detected correctly.
- Bug Fix: Rules: Rule Editor could crash the server if some of the condition or action fields were missing in the (customized) form.

**3.3b8** 21-Jun-00

- LDAP: the 'mail' attribute is now composed on-the-fly for records of the CommuniGateAccount objectClass.
- LDAP: the authentication methods are improved and documented.
- WebUser: the Directory Search page can now use the "internal CGatePro" names instead of the standard attribute names (i.e. `RealName` instead of `cn`).
- WebUser: Security: the Security Certificate (RFC2585) link is added to the Login page. See the Security section of the manual.
- WebUser: Non-ASCII Mailbox names are supported now.
- WebUser: format=flowed processing (RFC2646) is implemented.
- Router: the Add *name* to Non-Qualified Domain Names option is implemented.
- LOCAL: the Account Detail addressing Routing options are implemented.
- TLS: session recycler is implemented.
- TLS: the SSL 3.0/3.1 interaction is improved (interoperability with both Microsoft products and open source utilities).
- SNMP: TLS monitoring agents are implemented.
- Mailboxes: .mdir (MailDir) format is redesigned to store the number of message text lines in message file names. This should help some mailers (such as Netscape) correctly process messages with attachments retrieved from Unix servers.
- LIST: feed headers and trailers are correctly inserted into base64-encoded messages.
- Transfer: message with extremely long header fields (>100K) are now rejected with the SMTP, RPOP, PIPE, or POP (XTND XMIT) modules.
- Bug Fix: WebMail: in 3.3b7 2-byte charsets could result in infinite loops.
- Bug Fix: SMTP: the "relayHost" field was not always filled correctly resulting in garbage on the SMTP Monitoring pages.
- Bug Fix: IMAP: in 3.3b6-b7 the LIST "%" command might not list folders that were not Mailboxes at the same

time.

**3.3b7** 12-Jun-00

- SMTP: RFC2645 (ATRN) is implemented in both server and client modes to support mail delivery to hosts with dynamic IP addresses.
- WebUser: texts using Japanese ISO-2022-jp charsets should be displayed correctly now.
- LOCAL: all-domain aliases are case-insensitive now.
- LOCAL: the new Alert Text option allows you to specify the "over the quota" alert message text.
- TLS: the SSL 3.0/3.1 interaction is improved (interoperability with open source utilities).
- Domain: the 2-Letter 2-Level Domain Hashing now provides a workaround for accounts with 1-letter names and for accounts that have the dot symbol as the second symbol of their names.
- Bug Fix: IMAP: in 3.3b6 the LIST commands with non-empty prefixes returned incorrect results.
- Bug Fix: IMAP: in 3.3b6 the SELECT command could improperly capitalize the inbox Mailbox name, resulting in duplicated (INBOX and inbox) records in the account.info file.
- Bug Fix: OS/400 version improperly passed parameters to external tasks.

**3.3b6** 01-Jun-00

- Linux/StrongARM version is released.
- Directory: Access Rights (ACLs) are implemented.
- Directory: Browser and Access Right WebAdmin pages are implemented.
- Alerts: automatic "account is over quota" alerts are implemented. See the SysAdmin->Alerts section of the manual.
- LIST: the Digest generator has been modified to fit the RFC1153 requirements.
- SMTP: secure connections with forwarding servers are supported now.
- Mailboxes: Mailbox aliases are transparent now (included into Mailbox hierarchy views).
- WebUser: longer, alpha-numeric "session passwords" are used now.
- WebUser: the DirectoryFields arrays are added to the Strings.data file.
- IMAP: SNMP monitoring agents are implemented.
- WebUser Interface: the `Files` realm is implemented to provide access to arbitrary files in the WebUser directory.
- SysAdmin: the --noLockFile option is implemented.
- CLI: GetAccountLocation, GetServerRules, and SetServerRules commands are implemented.
- Bug Fix: IMAP: 3.3b5 did not place the quote marks around the "boundary" parameter value in the BODYSTRUCTURE response.
- Bug Fix: Directory: in 3.3b3-5 local Storage Units could generate the "non-text data" errors.
- Bug Fix: CLI: GetAccountAliases command could crash the server is the specified domain did not exist.
- Bug Fix: ACAP: multi-level searches in Dictionary DataSets could cause synchronization deadlocks.
- Bug Fix: ACAP: MODTIME responses for the SEARCH command were returned untagged.
- Bug Fix: ACAP: the optional metadata list in the RETURN clause of the SEARCH command was not properly parsed.

**3.3b5** 01-May-00

- Mailboxes: Mailbox Aliases are implemented (see the Objects->Mailboxes section of the manual). Mailbox aliases can be used to provide access to foreign Mailboxes for IMAP clients (such as MS Outlook / OE) that cannot process foreign Mailbox names in the Mailbox Subscription lists.
- WebAdmin: the Account Template page now includes the Initial Mailbox Aliases panel.
- Directory: Remove and Relocate Storage Unit operations are implemented.
- IMAP, ACAP: the output buffering method has been changed.
- Domains: the Generate Index option is implemented. It can be used to decrease the restart time for domains with 100,000+ accounts.
- Rules: the Current Day conditions are implemented.

Bug Fix: SMTP: the 3.3b4 version crashed if the Send Encrypted to Domains setting contained an empty list.

**3.3b4** 24-Apr-00

- Security: SSL/TLS client-side connections are implemented.
- SMTP: the Send Encrypted option is implemented to support server<->server encrypted message transfer.
- WebUser: attachment file names and HTML text portions are MIME-decoded and (optionally) converted to UTF-8 now.
- WebUser: the Use MIME for Headers option is implemented. When this option is selected, the Subject, To, Cc, and Bcc header fields containing non-ASCII symbols are MIME-encoded.
- WebUser: Alerts, Bye, Hello, List, ListArchive, Public, Rules, Rule, RPOP, Subscribers, Subscription pages now include the '=' (charset) macro symbols.
- WebUser: the '^$' macro (domain name) now works in all Account pages.
- WebUser: all ISO-8859-x and windows-125x charsets are now supported in the UTF-8 mode.
- Foundation: DNS addresses are now correctly retrieved from the Windows 2000 Registry.
- WebAdmin: clicking the Refresh button on the General Settings page tells the Server to re-read the DNS addresses from the OS.
- LIST: the First Digest At setting processing has been changed (see the LIST module manual).
- Rules: the Current Date and Time of Day Rule conditions are implemented.
- Rules: the Write To Log action is implemented.
- Bug Fix: WebUser: the "New" counters on the Mailboxes page were not updated properly.

**3.3b3** 17-Apr-00

- Directory: multiple Sub-tree storage units are implemented.
- Directory: schema editor is implemented.
- WebUser: UTF8 Mode Setting is implemented. The Settings.html and Compose.html pages have been modified.
- WebUser: the Settings.html, RPOP.html pages have been changed (^m has been added).
- WebUser: Compose: now the From, To, Cc, and Reply-To original letter headers are MIME-decoded first.
- Mailboxes: BSD Mailbox Manager now checks the size of text lines in new messages.
- Admin: the WebSite, BasicAccountSettings, and WebUserSettings Domain Access Rights are implemented.
- SNMP: WebUser monitoring agents are implemented.
- SNMP: POP module totaling-type reporting is implemented.
- Directory Integration: the special-case dc RDN attribute is supported now.
- Rules: comparison operations now ignore the surrounding angle brackets in the envelope (Recipient, Return-Path) addresses.
- Manual: the WebMail section is created to contain all user-level information about the WebUser Interface.
- Manual: the HTTP and HowTo sections are updated to provide the information about username.domain.dom Personal Web Site URLs.
- Bug Fix: the Header-field Rule condition did not process wildcard symbols correctly.

**3.3b2** 02-Apr-00

- Linux/Sparc version is released.
- SNMP: StalkerMIB ObjectID has been changed to 5678 - the IANA-registered Stalker ObjectID.
- Directory: the Directory Manager is implemented.
- Directory: the File-based Directories are implemented.
- Directory: the LDAP-based Directories are implemented (with caching).
- Directory: the .tdb and .ldb DataBase Managers are being phased out.
- Domains: the Directory Integration Settings are implemented.
- LDAP: implementation is based on the Directory Manager now.
- LDAP: RFC2254 is implemented.
- Account/Domain: the Directory-based Domains are partially implemented.
- RPOP: support for remote POP servers that do not return the number of messages in the initial UIDL response.

Account/Domain: the "Mobile" setting is enforced now: domains and account w/o that Access Mode cannot be accessed from non-client IP addresses.

- SMTP: Log reporting of relayed messages is improved.
- HTTP: WebUser port handling has been changed to support domain-style Personal Web Sites (i.e. http://user.domain.com can be processed as http://domain.com/~user/ now).
- WebUser: Korean, Chinese, and UTF-8 encodings are supported now.
- WebUser: the Mailbox page now remembers the filter and search strings and the position in the selected message set.
- WebUser: the Empty Trash button is now implemented on the Mailboxes Page.
- DEQUEUER: engine is redesigned and converted into a multi-threaded one.
- SMTP: statistical data (available via SNMP) is extended.
- SMTP: the SIZE EHLO response does not include the '=' sign now.
- IMAP: some responses are sent in batches now. This should improve performance over SSL connections.

**3.3b1** 03-Mar-00

- SNMP agent is implemented.
- Groups are implemented. GROUP-related CLI commands are implemented.
- Forwarders are implemented. FORWARDER-related CLI commands are implemented.
- LIST: the Special posting mode is implemented to allow subscribing of a mailing list to some other list.
- SMTP: the AUTH=address Mail From parameter is now supported.
- SMTP: batch-splitting for messages with multiple recipients in one domain is implemented.
- Bug Fix: CLUSTER: slave nodes refused to connect to the controller if one of the shared domains had the WebCache option disabled.
- Bug Fix: SMTP, RPOP: if a remote server was misconfigured, garbage left in the response buffers could cause problems when connecting to other servers serving that remote domain.

---

**3.2.4** 14-Feb-00

- HTTP: document name extensions are now converted into lowercase before checking them against the file extensions specified in the MIME Type table.
- The sendmail and mail utilities now use the backslash separators to compose files names on Win32 platforms.
- CLUSTER: SMTP AUTH is now supported for accounts in shared domains.
- Security: serverKeyExchange protocol is implemented to let "exportable" browsers connect to the domains that have strong (>512 bit) keys.
- Security: now domain aliases can be specified for Certificates and CSRs.
- Bug Fix: updating account settings via the WebAdmin Interface did not always update the Central Directory.
- Bug Fix: WebSites: references to Personal Web sites for users that had dot (.) symbols in their account names did not work without a trailing slash.
- Bug Fix: IMAP: COPYUID reporting was incorrect when Mailbox messages were copied into the same Mailbox.
- Bug Fix: in 3.2.3 the Mailbox#Username scheme did not work for POP APOP login.

**3.2.3** 28-Jan-00

- RULES: the Additional headers set with the Server-Wide Rules are now used with the LOCAL delivery and SMTP modules (before, those headers could be used only for Store and Redirect operations started from the same set of Server-Wide Rules).
- Foundation: socket-reading routines have been changed to avoid race conditions.
- WebUser: the Inbox name in the Mailbox Subscription lists is now processed with case-insensitivity.
- CLUSTER: the Authenticate then Relay feature now works for POP and PWD clients connecting via front-end servers.
- Migration: the MoveIMAPMail program now understands literals in Mailbox names and can copy Mailboxes

with names containing space characters.

- Bug Fix: CLUSTER: the front-end addresses were used instead of the client addresses when a client was closing connections and the client address was being removed from the TempClients table.
- Bug Fix: WebUser: some of the Mailbox view parameters were not processed correctly if they were set to the "default" values.
- Bug Fix: WinNT: passwords verification using NT accounts (LogonUser function) could crash the server or result (later) in various errors if an incorrect password was supplied.

**3.2.2** 18-Jan-00

- SMTP: the Advertise AUTH option can be set to Non-Clients now.
- LIST: the Who can browse option can be set to "Clients" now.
- Bug Fix: CLUSTER: the PWD module could not connect to back-end servers.

**3.2.1** 14-Jan-00

- Security: the TLS module 128-bit encryption is enabled.
- Bug Fix: CLUSTER: some admin operations did not convert account names to lowercase resulting in Controller de-synchronization and auto-shutdown.

**3.2** 12-Jan-00

- Additional Mailing List Licenses are supported now.
- Bug Fix: certificates with long serial numbers were not accepted.

**3.2b9** 26-Dec-99

- Foundation: passing environment parameters now works on the WIN32 platforms, too.
- HTTP: CGI applications can receive POST/PUT request data now.
- HTTP: Settings->WebUser page now allows to enter "starter" programs for CGI applications with specified extensions (so Perl scripts can now be used on Win32 platforms, too).
- Admin: the "Mobile" access mode is renamed into the "Relay" (existing settings should be updated automatically).
- Admin: the new "Mobile" access mode is added.
- Admin: the CertificateType Domain access right is added (when disabled, the domain admin cannot modify the Domain Security settings).
- SMTP: the alternative delay interval settings are implemented.
- IMAP: the size limitation for the APPEND command has been removed.
- LOCAL: the All-Domain Aliases are implemented (see the Local Delivery section of the manual).
- The {NS-MTA-MD5}-type digested passwords are supported to simplify migration from Post.Office servers. See the Migration section for the details.
- WebUser: the https:// URLS are now detected in plain-text message bodies and converted into clickable links.
- WebUser: the speed of composing HTML pages for large Mailboxes is increased.
- Bug fix: HTTP: a channel could crash when returning CGI results longer than 1 pipe block.

**3.2b8** 15-Dec-99

- Foundation: the synchronization routines have been modified on all platforms to improve performance under a heavy load.
- Foundation: optimized timer in 3.2b6-7 could result in queue processing delays on multi-processor systems.
- Obscure: the Use Conservative Info Updates option is implemented. Can be disabled on heavy-loaded systems, especially if accounts are stored on an NFS server.
- The Do Not Redirect Automatic Messages option is added to the Simplified Redirection Rule.
- The {MD5} and {SHA} digested passwords are supported to simplify migration from Netscape and software.com servers. See the Migration section for the details.

- Secondary Domain names are explicitly converted to small letters now.
- The QUEUE Log records for new messages now include the message Return-Path.
- WebUser: when an open message is moved to a different Mailbox and the account Delete mode is not "Mark", the original message is deleted.
- Rules: the Each/Any Recipient conditions are renamed into the Each/Any Route conditions, and the new Each/Any Recipient conditions are implemented (see the Rules section for the details).
- Bug Fix: domain renaming involving Central Directory updates could cause crashes.
- Bug Fix: in 3.2b6-7, changing the poll period of an RPOP record could result in RPOP record disappearing.

**3.2b7** 07-Dec-99

- Many routines are switched from linear search algorithms to binary search ones.
- Security: the per-Domain Custom certificates are implemented.
- WebUser: the "Recover passwords via E-mail" functionality is implemented: the Password.html page is added to retrieve forgotten passwords. A link to that page (appearing on the "Incorrect Password" error) is added to the Login (default.html) page. The Settings.html page has been modified to let users change the Recovery Address setting.
- Admin: the speed of importing accounts with aliases is improved.
- SMTP: AUTH command is accepted from Client IP Addresses only if the Grant Access to Clients Only option is enabled.
- RFC2595 is implemented: IMAP, ACAP and POP modules support the STARTTLS/STLS command now.
- Bug Fix: Rules: the Return-Path condition did not work in 3.2b6.
- Bug Fix: RPOP: double-unlocking of the RPOP queue could generate exceptions in 3.2b6.
- Bug Fix: the 3.1-3.2 versions did not properly clear the WebUser cache.

**3.2b6** 29-Nov-99

- The Dynamic Cluster code is included into the mainstream releases.
- SMTP: recipient address checking has been changed to accommodate advanced Router techniques (routing to non-standard ports, bypassing MX records, etc.)
- DNR: now TCP connections are used to read extra-large DNS responses.
- CLI: the GETDOMAINALERTS and SETDOMAINALERTS commands are implemented.
- CLI: the SETDOMAIN, SETACCOUNT, SETACCOUNTDEFAULTS, SETACCOUNTTEMPLATE commands are implemented.
- CLI: the GETACCOUNTRPOP and SETACCOUNTRPOP commands are implemented.
- CLI: the CREATESHAREDDOMAIN command is implemented.
- CLI: the GETWEBUSER, SETWEBUSER, GETWEBUSERDEFAULTS, and SETWEBUSERDEFAULTS commands are implemented.
- CLI: the GETWEBUSERINTERFACE, PUTWEBUSERINTERFACE, DELETEWEBUSERINTERFACE, LISTWEBUSERINTERFACE, and CLEARWEBUSERCACHE commands are implemented.
- CLI: the LISTMAILBOXES command is implemented.
- CLI: the ROUTE command output format has changed.
- ROUTER: the ".here" domain names suffix is supported now.
- Admin: the Refresh button is added to the General Settings page.
- WebAdmin: the WebAdmin interface and WebUser interface Editors pages have been moved inside the WebAdmin/Accounts directory.
- WebUser: custom domain files cache speed is improved.
- Rules: the "in" and "not in" condition operations are implemented.
- Bug Fix: PIPE: module routing was broken in 3.2betas.
- Bug Fix: the General Settings WebAdmin page did not update all its settings in 3.2betas.
- Bug Fix: the mobile client support (Temp Addresses) was broken since 3.2b2.
- Bug Fix: SMTP: the module could crash if the highest priority MX record for a recipient domain had a priority value >= 32768.
- Bug Fix: IMAP: the BODYSTRUCTURE response was not composed correctly for Content-type: message

subparts.

**3.2b5** 04-Nov-99

- IMAP: the fld-disp BODYSTRUCTURE extended data field output has been modified to provide compatibility with legacy IMAP clients.
- IMAP: the Literal format is used for ENVELOPE and BODYSTRUCTURE fields containing non-ASCII or special characters.
- SMTP: AUTHorization can be used to cleanup the "blacklisted" host flag.
- LIST: moderated subscribing mode is implemented.
- LIST: messages stored in the *listname*/request Mailbox now get the `X-List-Report` header.
- Admin: the CanAccessMailboxes domain administration Access Right is implemented.
- WebUser: letters/parts in the multipart/related format are supported now.
- WebUser: the "Open Mailbox" feature is added to the Subscribed Mailboxes page.
- Router: the default Relaying mode for Router records has been changed to `Relay` and the `NoRelay` tag is implemented. See the Router section for the details.
- Rules: the Any Recipient and Each Recipient conditions are now supported in the Server-Wide Rules.
- Bug Fix: very long input lines (>65K) sent to the service streams could cause server crashes.
- Bug Fix: memory leak in Domain Routing is fixed.
- Bug Fix: memory leak in WebUser Interface (Message viewer) is fixed.
- Bug Fix: WebUser: the Mailbox Refresh settings was not working correctly.

**3.2b4** 25-Oct-99

- RBL: now supporting RBL servers such as ORBS that return non-standard responses.
- WebSite: CGI scripts are now supported (use the Settings->WebSettings page to specify the directory where the the /cgi-bin/ program files are located).
- WebSite: the banner insertion routine is improved.
- WebUser: the Show HTML inline/in frame option is implemented.
- WebAdmin: the Domains page processing is improved (aliases, sorting).
- Bug Fix: if a Secure (SSL/TLS) connection failed on an early stage of negotiation, the server could crash.
- Bug Fix: in 3.2b1-3 domain aliases for the main domain could not be removed.
- Bug Fix: IMAP: the BODYSTRUCTURE response had a space between the "multipart" part descriptors and did not have spaces between multiple part-body attributes. This could cause a problem to the "pine" IMAP client.

**3.2b3** 18-Oct-99

- Domains: Reroute Mail to Unknown: the "*" symbol processing is improved to put quotes back into the original "quoted" addresses.
- WebUser and Rules: if a recipient E-mail address is specified without the domain part, the domain name of the current account is added to the envelope address.
- Security: the U-crpt password encryption is supported for Unix platforms. This is useful for moving account passwords from old servers. See the Security section for the details.
- Security: now empty CommuniGate Passwords cannot be used for logging in, but they can be updated if users have logged in using OS or External Authentication.
- WebUser: attachments processing on the Compose.html page has been changed (to allow localization and modification of max. number of attachments). Update your customized Compose.html pages.
- Admin: the UnixPassword column is now supported in the Account Loader import files.
- Bug Fix: in 2.3b1-2, the SMTP AUTH workaround for Netscape Mail servers was not working.
- Bug Fix: in a client using SMTP AUTH could crash the server.
- Bug Fix: an uninitialized variable problem could crash the PWD server (accessed via its TLS port).

**3.2b2** 10-Oct-99

- Account Management: storing Account Settings in Central Directory database is implemented.

- Cluster: frontend-backend Clustering for SSL/TLS connections is implemented.
- Cluster: the Directory-based Cluster Routing is implemented.
- RPOP: the Leave on Server option is implemented.
- LDAP: array-type database data elements can be returned now (as multi-value LDAP SET elements).
- DataBase: TextManager (.tdb) databases now support complex (array, dictionary) data elements.
- DataBase: LDAPManager (.ldb) databases now support array-type (multi-value) data elements.
- SMTP: the "Use Forwarding Server" processing mode has been changed.
- SMTP: the '#' and '@' ETRN parameter prefix symbols are accepted (and ignored) now.
- MIME: headers with "folded strings" are processed correctly now.
- Account, Domain, SMTP, POP: the Mobile "Account Service" setting is implemented. Check that this option is enabled for all your mobile users. This is an anti-spam measure useful for sites providing free WebMail services.
- BSD Mailbox: message flags updating is delayed now. This should increase performance on under heavy load, especially POP3 load.
- Temp File: the Recycle Temp File option is implemented (see Obscure settings); this should increase performance under heavy load.
- The "Mail to Unknown" setting now applies only to mail routing, not to account name (Access) routing.
- WebSite Uploading: application/x-macbinary format is decoded now, so MS Explorer on Mac can be used to upload Web files.
- Bug Fix: LDAP: in 3.2b1 BIND-ing as "anybody" (using an empty username string) could crash the server.
- Bug Fix: 3.2b1 Account Rules Web interface failed to create a new Rule.
- Bug Fix: SMTP: the "Relay for hosts we backup" option did not work correctly.
- Bug Fix: IMAP: the LIST "" "name/%/%" command did not work correctly (Netscape could not see nested mail folders).
- Bug Fix: the Central Directory "Who can Browse" option was reset to "Anyone" after server restart.
- Bug Fix: message envelope addresses containing 8-bit characters inside a q-string were rejected.

**3.2b1** 17-Sep-99

- SMTP, POP, IMAP prompt strings and responses can be customized now. See the SysAdmin section of the manual for the details.
- LDAP: the AddRecord, DeleteRecord, ModifyRecord, and ModifyDN operations are supported now; the LDAP user should be authenticated and should have the All Domains server access right to be able to use these operations with the Central Directory database.
- The LDAP-based DataBase manager is implemented. See the Data section of the manual for the details.
- ACCOUNT: now if the Secure Authentication is set to Required, clear text passwords are still accepted, if they are passed via a secure (SSL/TLS) connection.
- ACCOUNT: an external authenticator can be used for SASL-type authentication.
- WebAdmin: access to account WebUser settings is now provided via a link on the Account Settings page.
- WebUser: the WebUser Settings now have the sever-level and domain-level Defaults (same mechanism as used for the Account Settings). The Defaults can be set by a server/domain administrator using the links on the Default Account Settings pages.
- WebUser: the Settings page: the ^A macro has been changed to ^b and the ^L macro has been removed.
- WebUser: Personal WebSite management has been moved inside WebUser Session processing (no additional authentication required).
- WebAdmin: access to the Personal WebSite management is now provided from within the WebAdmin "Domains" realm (no additional authentication required).
- WebUser: the SaveDraft and Save Sent operations now save messages with the "seen" flag pre-set.
- RULES: additional headers specified for React/Reply messages can now contain macro symbols.
- RULES: additional headers specified for Reply messages can now contain To/Cc header fields.
- RULES: additional headers specified for React/Reply messages can now contain Bcc header fields.
- CLI: the GetAccountRules and SetAccountRules commands are implemented.
- Bug Fix: LIST: "account closing" house-keeping procedure could cause a deadlock in the entire kernel (via locking the Router).
- Bug Fix: WebAdmin: WebSession Monitor processor generated the "STObject refCount != 0" messages in the

OS system Log. This bug should not cause any problem.
- Bug Fix: the 3.1 LIST "cleanup" fix was not correct, unconfirmed subscribes still could be removed too early.
- Bug Fix: sometimes when the INBOX Mailbox was renamed, a new INBOX was not created.

---

**3.1** 01-Sep-99

- CLI: the UpdateModule command is added.
- CLI: the GetAccountInfo command is added.
- LIST: refs to the list Web Archive now point to the secure port if no unsecured one is enabled.
- WebAdmin: the Account Settings page now contains a link to the user Personal Web Site.
- Bug Fix: LIST: unconfirmed subscription requests were deleted from the subscriber list during each "cleanup" job, without the 2 days waiting interval specified in the manual. As a result some new subscribers had to send confirmation requests twice.

**3.1b9** 25-Aug-99

- WebUser/WebAdmin: image buttons are now accepted in multipart/form-data forms.
- WebAdmin: an administrator can now remove Alert messages.
- Domain: security options can be specified for "all"-type message distribution.
- WebUser: the mailing lists page now displays the list names in the sorted order.
- WebUser: the Via Trash and Immediate delete modes are implemented. The Delete Method setting is added to the WebUser Settings page.
- WebUser: several additional message strings were placed into the Strings.data file.
- WebUser: the Baltic, Arabic, and Chinese charsets are added.
- Rules: the Header Field condition operation is implemented.
- Bug Fix: Accounts Importing did not work in 3.1b8.
- Bug Fix: the 3.1b7-b8 version did not place the "trusted" flag into the Mirror-ed messages.
- Bug Fix: updating the RPOP accounts could crash the 3.1b8 version.
- Bug Fix: the WebAdmin SMTP Read Monitor could crash if it saw a stream going through a clean-up procedure.

**3.1b8** 23-Aug-99

- MIME headers decoder now converts the underscore sign into the space symbol.
- Accounts: the External Authenticator is implemented (see the Security section of the manual).
- WebUser: if the Subject field of the composed messages contains 8-bit characters, it is sent in the MIME-encoded form now.
- PIPE: message delivery is multi-threaded now.
- PIPE: the module now processes the Envelope-Ignore and the Envelope-ID fields in submitted messages.
- LIST: the Insert Feed Prefix after Reply Prefix option is implemented.
- Kernel: the thread-killing mode is implemented.
- Rules: the Redirect/Mirror/Forward operations now put "additional headers" into the generated messages.
- sendmail: the --, -t, and -V parameters are supported now; processing of the -f and -F parameters has been improved.
- Bug Fix: the server could crash when displaying the Queue WebAdmin Monitor page on a heavily loaded system.
- Bug Fix: the server could crash when trying to access a just-deleted Mailbox message if the message was the only one in that Mailbox.
- Bug Fix: 3.1b7 direct Mailbox addressing could crash the server.
- Bug Fix: 3.1b7-b6 could crash when inserting "hot links" into message HTML representations.
- Bug Fix: ACAP: comparison operations with the NIL operand could crash the server.

**3.1b7** 06-Aug-99

- Protection: multiple RBL Servers are supported now.

- SMTP: the name of the RBL server is now included into the rejection response.
- SMTP: unqualified recipient names are now qualified with a secondary domain name, if the SMTP connection is established to an IP address assigned to a secondary domain.
- SMTP: the initial prompt timeout for not-last-chance relays increased to 2 minutes.
- SMTP/LOCAL: the Host Monitor pages now allow a system operator to specify the error code for the Reject operation.
- The Mirror-to operation (applied to queued messages) now preserves the original Return-Path envelope address.
- Account: the New MailBoxes Type setting is implemented.
- Account: Mailbox case sensitivity is enforced when running under MS Windows.
- LOCAL Delivery: Direct Mailbox Addressing is implemented.
- POP: Direct Mailbox Addressing is supported.
- Mailbox storage: "common base" storage is implemented for "Directory" Mailboxes (when a queued message is being delivered to many directory-type Mailboxes, only one OS file is created).
- WebUser: the AutoWrap option is implemented. The "wrap=virtual" option is added to the Compose.html page.
- WebUser: the charset (^=) macro combination is added to service pages (mailboxes.html, settings.html, etc.)
- Redirecting/Forwarding: when multiple addresses are used, all addresses are included into the "To" header.
- Bug Fix: the new HTTP Redirect realm did not always set the text/html Content-type for its response.
- Bug Fix: in 3.1b6, updating the Router table on a Unix server inserted "return" symbols into the Table text.
- Bug Fix: WebUser: references to foreign Mailboxes (return jumps) were composed incorrectly.

**3.1b6** 02-Aug-99

- Central Directory: the Who Can Browse setting is implemented.
- Access: the Server-wide and Domain-Wide Alerts are implemented.
- LIST: the Keep To And Cc option is implemented.
- WebUser: web access to Central Directory is implemented (for both Logged in and Guest users).
- WebUser: the mailto "hot links" seen within a WebUser session open the WebUser Composer now.
- WebUser: now http "hot links" are processed via a Redirecting page to prevent Referer-based account cracking.
- the `sendmail` legacy mailer emulator is added.
- Bug Fix: account caching mechanism had a synchronization bug, which could cause memory corruption problems, especially under a heavy load.
- Bug Fix: on some PPC platforms the domain alias data was not processed correctly.
- Bug Fix: WebUser: references to foreign Mailboxes were composed incorrectly.

**3.1b5** 19-Jul-99

- Domain: the Server OS User Name picture setting is implemented.
- Account: now OS passwords can be used for accounts in all domains (using the OS User Name picture settings).
- Rules: the FingerNotify action (useful for the NotifyMail® users) is implemented.
- HTTP: now the Date and Last-Modified HTTP headers are returned with static objects.
- WebUser: file extensions -> MIME association table is implemented (WebSession Settings).
- WebUser: now all message text components are displayed in the "wrappable" format, rather then in the "pre-formatted" format.
- WebUser: the Received header with the user current IP address in included into all composed messages.
- WebUser: the URLs in message bodies are presented as active URL links.
- WebUser: an alternative WebUser Session access method (/Session/ID/xx instead of /Account/xx?SID=ID) is implemented; the entrance and several default pages have been switched to use this method, and the "Message.html" portion processing is switched to use this method to avoid problems with file downloading using MS Internet Explorer.
- WebUser: Greek and Central European charsets are added. Charsets are now displayed by name rather than by their MIME codes.
- SMTP: messages received via authenticated connections are marked as "submitted by authenticated users".
- SMTP: the maximum number of parallel sending channels is automatically opened when the domain queue contains too many messages.

- LIST: duplicate envelope addresses are processed as one address now.
- LIST: the Verify Owner setting is implemented.
- LIST: the current number of subscribers is displayed on the List Monitor and Account Setting pages (when that number is available).
- LIST: the Notify Owner option is implemented.
- LIST: now the message body charset is specified in List service messages (Warnings, Welcome, etc.)
- SMTP: the ETRN command now applies the server Router to its argument.
- SMTP: the Hold Mail for Domains option is implemented.
- Account: the Add Mail Trailer and Add Web Banner settings are implemented.
- Rules: when a Reaction/Reply message is composed, the original subject text is MIME-decoded before substituting.
- Bug Fix: modifying Server-Wide Rules on a heavily-loaded server could result in a crash.
- Bug Fix: if the Personal Web Site size setting was set to Unlimited, the Web Site current file number was calculated incorrectly.
- Bug Fix: the 3.1b4 version failed to route addresses with IP-type domain parts.

**3.1b4** 07-Jul-99

- Kernel: the Crash Recovery General Setting is implemented (obsoletes the --NoCatch and --NoExceptions command line options).
- WebInterface: the CommuniGate Pro Guide are now shipped with the software and all the links in WebAdmin and WebUser pages are changed to the local server copy of the Guide.
- Rules: the Redirect All Mail simplified Rule is implemented.
- Accounts: files used for New Accounts Import can specify all account settings, as well as Central Directory supplementary fields.
- WebUser: access to the Index.html Personal Web Site file requires authorization now.
- WebUser: the Index.html Personal Web Site file format is customizable now; the page form fields allow an authorized user to upload files to that Personal Web Site, and to remove files from that site.
- WebUser: caching is optimized.
- WebUser: form-based page Editor for WebUser pages is implemented.
- WebAdmin: HTTP and form-based page Editor for WebAdmin pages is implemented.
- WebUser/Domain: The WebSite Banner Domain settings can be used to specify some HTML codes to insert into all HTML files retrieved from the domain Personal Web Sites.
- WebUser/WebAdmin: graphical (image) buttons are supported now.
- WebUser: "window" browsing for Mailboxes and mail list archives is implemented (i.e. when there are more messages that can be displayed on the page, the user can use the next/previous buttons).
- CLI: the GetAccount * command (available to all users) in implemented.
- BSD Mailbox: a Mailbox with a damaged last line (w/o EOL) cannot be opened now (manual Mailbox correction is required).
- Logs: time stamp composing algorithm has been changed.
- Router: module-level routing is optimized.
- Bug Fix: account passwords set via the WebAdmin Interface were encrypted even if the encryption setting was set to "clear text".
- Bug Fix: MailDir-type accounts were not processed correctly after server restart.
- Bug Fix: the SSL/TLS module could report "wrong version number" because of a buffer processing bug.

**3.1b3** 30-Jun-99

- Kernel: X.509 certificate routines are implemented.
- Kernel: BigNumber library is implemented.
- Kernel: Cryptography library is implemented.
- Kernel: SSL 3.0 and portions of TLS 1.0 security protocols are implemented.
- Listener: Local Address binding is implemented (to allow the server to accept connections on a selected local IP address only).

- Listener: multi-socket listeners are implemented (the server can now accept connections on several port numbers and/or several local IP addresses).
- Listener: individual Remote IP Address restrictions are implemented.
- SMTP: the STARTTLS extension (secure communication) is implemented.
- SASL: the LOGIN method accepts initial parameters now (should eliminate the AUTH problem when getting mail from Netscape Servers).
- LDAP: during an initial install, the LDAP module is configured to accept clear text connections on the TCP port 389, and secure connections - on the TCP port 636.
- LDAP: the BIND operation processing has been changed to provide compatibility with Netscape and Microsoft clients.
- IMAP: during an initial install, the IMAP module is configured to accept clear text connections on the TCP port 143, and secure connections - on the TCP port 993.
- IMAP: the UID messages set specifications n:* where n is larger than the largest UID in the Mailbox return empty sets now.
- MIME: searching inside MIME-encoded headers is implemented.
- PIPE: support for Envelope-To: headers is implemented.
- RPOP: retrieving UDWA messages has been improved.
- WebUser: the Cache Files option and the Flush Cache button are added to the Domain Settings.
- WebUser: the MailboxViewRegularHeader and MailboxViewHilitedHeader elements are added to the Strings.data file.
- Bug Fix: the Reply/React Rule operation could place garbage into the From: field of the generated messages.
- Bug Fix: 3.1b2 versions could crash when logging a Rule redirect/forward./mirror action.
- Bug Fix: 3.1b1-2 versions could crash when redirecting messages using a Server-wide Rule.

**3.1b2** 09-Jun-99

- WebAdmin: Account Settings, Template and Defaults have separated HTML page files now.
- Admin: the Account Template can now specify the additional Mailboxes to create for all new accounts and the initial Mailbox subscriptions for new accounts.
- WebUser: LIST browser: next/previous message switching mechanism has been changed to avoid an intermediate jump page. The old mechanism is still supported for compatibility with old customized List Browser pages.
- WebUser: the `Cache-Control: no-cache` HTTP attribute is added to various WebUser interface pages.
- WebUser: the FORM element in the Mailbox.html page has been modified to avoid various problems (like inability to undelete a just-deleted message).
- WebUser Mailbox browser: the From field is substituted with the To field for the Mailboxes specified as Sent and Drafts Mailboxes.
- WebUser: the reply/forwarded message Composer MIME-decodes the original message Subject.
- LIST: message subjects are MIME-decoded and excessive Re:/Re> prefix strings are removed when messages are inserted into a digest and/or distributed in the feed mode.
- Bug Fix: in 3.1b1 on most Unix platforms the account Rules page always showed the actions allowed for the Filter Only level.
- Bug Fix: in 3.1b1, generating a digest for a list without any digest subscriber could crash the server.
- Bug Fix: WebUser: the Save As Draft operation could place garbage into the message return-path field.

**3.1b1** 07-Jun-99

- Admin: the Account password encryption setting is implemented.
- Rules: the multi-level Rule specifying rights are implemented.
- WebUser: all attachments (including text files) are sent encoded now.
- HTTP: the Roaming support for Netscape is implemented.
- HTTP: the MOVE HTTP method is implemented for personal Web sites.
- LIST: Settings Web pages have been changed: the Archive/Digest settings have been moved to a separate page.
- LIST: The Digest Format setting is implemented.

- LIST: postings from "non-human" sources are rejected now.
- LIST: the `confirm` operation is implemented.
- Bug Fix: WebUser: displaying a Message page for a just deleted message could crash the Server.

---

**3.0.1** 30-May-99

- Relay restrictions: Mobile users support is improved (see the Protection section).
- SMTP: relay rejection code is changed to 4xx if the Mail From: address is routed to a local account address (this makes the "use POP, then send mail" method more useful for mobile users).
- Rules: the messages generated with automatic Redirect/Forward/Mirror rules get an additional `Received` header now; this stops "inside-the-server" mail redirection loops.
- MIME: charset name processing is case-insensitive now.
- WebUser: the MarkAll button is added to the Mailbox pages.
- WebUser: the Messages To Display option is added to the settings (it is used for account mailing lists, too).
- WebUser: attachment file name processing is improved.
- WebUser: text attachments are displayed as regular attachments, not as inline components.
- WebUser: the Info parameter can be used on the Mailbox pages now.
- LDAP: the Make LDAP Server Slower option is implemented (to avoid Netscape crashes).
- Bug Fix: LDAP server could crash when processing malformed search requests.
- Bug Fix: MIME: on RISC platforms, the destructor could cause stack overflow when releasing MIME structures with several thousand elements (as in WebAdmin HTML forms for large subscriber lists).

**3.0** 24-May-99

- Migration: the `MovePOPMail`, `MoveIMAPMail`, and `MoveAccounts` programs are included into the package. See the Migration section for the details.
- IMAP: the Envelope fields Sender and Reply-To are now defaulted to the From field as required by RFC2060.
- LIST: the unsubscribe and mode change operations can be now performed "silently". The "posting prohibited" policy can now be set as default for new subscribers.
- LDAP: if the "search root" includes the `c=country` string and the Central Directory does not contain the "c" data, the "c=country" filter is removed (to avoid problems with MS Outlook).
- WebUser: the "Edit Draft" operation is implemented.
- CLI: the ROUTE command is implemented.
- Bug Fix: LDAP: the "strictly equals" string operation parser could crash the server.

**3.0b8** 16-May-99

- IMAP: the Fetch BODYSTRUCTURE responses now include the Content-ID and the Disposition extension fields.
- Account: the Accept Mail To All option is implemented.
- WebAdmin: the WebAdmin directory in a secondary domain directory can be used to customize that domain administrator WebAdmin Interface.
- Bug Fix: IMAP: Fetch ENVELOPE parsing routine could address out-of-buffer data.
- Bug Fix: BSD Mailboxes: message saving and Mailbox parsing routines could read data beyond buffer boundaries, causing crashes on some platforms (FreeBSD).
- Bug Fix: the Mail for Unknown parameter did not work in 3.0b7 version.
- Bug Fix: changing the RPOP module settings could crash the server in 3.0b5-7 versions.

**3.0b7** 10-May-99

- Domain/Accounts: the concept of Default Settings is introduced (see the Accounts and Domains manual sections for the details.
- DomainAdmin: now the system administrator can specify all operations the domain administrator is allowed to do.

- Admin: now the system administrator should have only the "AllDomains" right, not the Master Right to specify access (domain administration) rights for the accounts in non-main domains.
- WebAdmin: the Domain Administration entrance page (accesses via http://domainname:8010) is unprotected now, and it contains a link to the protected domain administration realm.
- CLI: SetAccountTemplate command is changed into UpdateAccountTemplate.
- CLI: GetAccountDefaults/UpdateAccountDefaults operations are implemented.
- CLI: now the special value `default` can be used in the UpdateAccount and UpdateDomain commands.
- PIPE: the Processing Time Out option is implemented.
- LOCAL: the RFC822 field name for storing envelope recipients can be customized now.
- WebUser: the Composer component now selects the proper charset and adds the charset parameter to composed messages.
- PWD: prompt format and login processing has been changed to support the APOP authentication method.
- Bug Fix: QUEUE: Non-ASCII characters in envelope addresses are processed correctly now.
- Bug Fix: Domains: a domain with non-flat foldering could not be removed.
- Bug Fix: WebUser: the Refresh Mailbox rate was not properly stored in the Viewer settings.
- Bug Fix: DOMAIN: domain aliases were not processed correctly when a domain was renamed or removed.

**3.0b6** 28-Apr-99

- Domain/Account: the Enabled Services options are implemented.
- LOCAL: mail to domains/accounts with the disabled Mail service is suspended in queue now.
- LOCAL: now mail to "almost full" accounts can be suspended in Queue.
- LOCAL: WebAdmin monitor for waiting accounts and for account queues is implemented.
- WebAdmin: now administrators can use the HOST Monitor pages to reject all messages waiting in the Host queue.
- HTTP: internal caching scheme has been changed.
- Domain: RPOP accounts limit is implemented.
- Domain: zero limits can be specified for domain accounts, mailing lists and RPOP accounts.
- WebAdmin/WebUser: Mailing Lists pages are customizable now.
- WebAdmin: now domain administrator access is available for domains without A-records (and for domains that have A-records pointing to a different server). See the HTTP module section for the details (new URL schema).
- WebUser/Archive: preferred charset is used to view messages w/o any explicitly specified charset.
- CLI: the DLACNT alias of the DeleteAccount command did not work.
- CLI: the RenameAccount, RenameList, and RenameDomain command syntax has been changed.
- CLI: the ListLists command has been added.
- WebUser: the name of the HTML page to display when a user logs in is customizable now (stored in the Strings.data file).
- HTTP: a new WebUser setting allows you to specify a prefix for user personal Web Site URLs.
- Bug Fix: search operations in the domain alias database were case-sensitive.

**3.0b5** 16-Apr-99

- Security: the CommuniGate Pro passwords are stored in an encrypted format now.
- CLI: mailing list manipulation commands have been added.
- WebUser/WebAdmin: Routing of Domain names has been changed (see the HTTP/Routing section for the details).
- RPOP: the Last info field has been added to individual RPOP setting tables.
- RPOP: the Allow Self-Poll option is implemented.
- RPOP: WebAdmin Monitoring is improved.
- RPOP: Retrieval via non-standard ports (servername:port) is implemented.
- LIST: the Preferred charset and Reject Non-Matching Charset options are implemented.
- LIST: the fields in a digest Table of Contents are MIME-decoded.
- LIST: the charset is now specified for the TOC part.
- WebUser: the Preferred charset option is implemented.

- WebUser: Mailbox displaying algorithms/formats have been improved.
- Web/MIME: Message header fields encoded in the MIME format are decoded now..
- SMTP: relay selection algorithm is changed to provide better connectivity to slow/remote sites.
- SMTP: a space after ":" is removed from the Mail From: and Rcpt To: commands. Some broken mail servers do not accept SMTP commands with a space after the colon sign.
- SMTP: sending to non-standard SMTP ports is implemented.
- POP: locking model has changed to avoid problems with broken connections.
- The Aliases database is renamed into the DomainAliases database.
- WebUser: some strings have been moved to the String.data file.
- WebUser: an option specifying the default sorting order is added.
- WebUser: an option specifying the Mailbox refresh rate is added.
- WebAdmin: last access IP address is displayed in the Account List now.
- Monitor synchronization algorithms have been optimized.
- Windows NT: both local OS accounts and accounts in the trusted domains can now be used for OS-based authentication.

**3.0b4** 09-Apr-99

- Security: if a password check fails, client processing is suspended for 2 seconds now.
- CLI: GetAccountTemplate and SetAccountTemplate operations are implemented.
- CLI: GetAccountRights and SetAccountRights operations are implemented.
- Foundation: external task starting is modified to address thread-safety issues on the Solaris, FreeBSD, and Linux platforms.
- Bug Fix: routing of listname-request and listname-admin addresses was broken in 3.0b1-3 versions.
- Bug Fix: LDAP "substring searches" could crash the 3.0b1-3 versions.
- Bug Fix: the 3.0b3 version allowed unrestricted access to accounts that had the "use OS password" option set, if no user with the account name was registered with the server OS.

**3.0b3** 07-Apr-99

- WebAdmin: WebUser Monitor and WebUser Settings pages are implemented.
- WebAdmin: PIPE Monitor is implemented.
- IMAP: message-set processing changed (for Eudora compatibility).
- Kernel: backslash processing in RFC822 comments has been fixed.
- Foundation: a workaround for Solaris DST (Daylight Saving Time) bug.
- Bug Fix: IMAP processing in 3.0b2 could enter an infinite loop.
- Bug Fix: 3.0b1-b2 on some platforms failed to rename/remove accounts.
- Bug Fix: launching external tasks (Rules, PIPE) could lock.

**3.0b2** 02-Apr-99

- Major Foundation Library redesign, many components modified to take the advantage of the new Foundation classes.
- SMTP: the Message Recipients Limit option is implemented.
- SMTP/Router: the spamtrap address processing is implemented.
- SMTP: envelope processing is improved.
- SMTP: the ORCPT parameter is passed/generated when mail is relayed.
- RPOP: the Minimum Poll Period for Users option is implemented.
- RPOP: the Maximum Poll Accounts per User option is implemented.
- WebUser: address book entries sorting is implemented.
- WebUser: mailing list archives URL schema has changed.
- LIST: the Only Subscribers list archive Browse mode is implemented.
- CLI: GetAccountAliases and SetAccountAliases commands are implemented.
- CLI: GetDomainAliases and SetDomainAliases commands are implemented.

Bug Fix: the WebUser time-outer thread was not correctly synchronized with access threads; this could cause server crashes.

- Bug Fix: account removal operations could fail because the Personal WebSite files were not removed automatically.
- Bug Fix: Web pages uploading was broken in 3.0b1.

**3.0b1** 26-Mar-99

- Cluster: Cluster support Settings are implemented.
- Cluster: POP, ACAP, IMAP, and PWD cluster support is implemented.
- Cluster: Personal Web Site cluster support is implemented.
- Cluster: WebUser Interface cluster support is implemented.
- Account Prefs: Address Books are implemented.
- WebUser: access to account AddressBooks is implemented.
- WebUser: Save Sent Messages and Save as Draft options are implemented.
- Domain Aliases are implemented.
- PIPE: non-empty stderr output is now used in the error messages generated when processing ended with a non-zero code.
- Redirect: the automatically (via Rules) redirected messages have non-NULL Return-Paths now (instead, the notify=never DSN is used for all generated recipient addresses).

# Summary

**3.5** 18-Dec-01
**Platforms**

- Linux/S390 version is released.
- Linux/M68K version is released.

**Domains, Accounts, Objects**

- Domains,WebUser Settings: non-ASCII attributes are supported now.
- Account Templates: the Initial Message text now can start with the [charset] prefix.
- DIRECTORY INTEGRATION: Server-Wide and Cluster-Wide settings are implemented.
- WebAdmin: Admin account Preferences processor has been changed.
- Migration: the MoveAccounts program and its parameters have been changed.
- Mailboxes: the "Redirected" message status is implemented.
- Groups: non-ASCII "real names" are supported now.
- Accounts: non-ASCII "real names" and "custom attributes" are supported now.
- Account Templates: non-ASCII "real names", "custom attributes", and Mailbox names are supported now.
- WebSite: nested folders are supported now. The WebAdmin WebSite.html and WebUser WebSite.html and UserSiteIndex.html files have been updated.
- Now Domain Administrators can control several Domains. See the SysAdmin section of the manual for more Details.

**Protection**

- The Banned Header and Body line settings are implemented. See the Protection section of the manual.
- The Client By Name option is implemented.
- The Unblacklistable (WhiteHole) Addresses list is implemented.
- RBL: RBL responses in the 127.1.x.x range are now recognized as "blacklist it" responses.

**Security**

- The login-disabling options protecting Accounts from Password Attacks are implemented.
- Now Certificate Signing Requests can be generated and new Certificates can be set without prior removing of an existing certificate.
- The SASL NTLM method now works with Macintosh versions of Microsoft products (Outlook Express, Entourage).
- The SASL NTLM authentication method is supported now (this method allows you to use the "Secure Password Authentication" option in Microsoft products).

## DNR

- The "search PTR records" operation is implemented.
- The new Custom setting allows an administrator to specify DNS Server addresses explicitly. See the SysAdmin section for the details.
- Additional settings are implemented.
- Requests to RBL servers are sent "quickly", so if an RBL server is down incoming SMTP connections do not time-out.

## WebAdmin

- Preferences now work for Secondary Domain Administrators, too.
- The Charset parameter has been added to the Administrator Preferences.
- Alerts are stored using the UTF-8 charset.
- The /MainAdmin/ realm is implemented (see the HTTP section of the manual).
- Log settings can now be changed only if the Administrator has the CanModifySettings Server Access rights.
- Additional Monitor Access Rights are implemented.

## WebUser

- WebSkins are implemented. Now all domains that do not have files in the Account subfolder of the old custom WebUser Interface directory use the new Skins Interface.
- The "Trash" Mailbox can now be replaced with a Mailbox alias (useful to create a "shared Trash" Mailbox for several accounts).
- The "Redirected" message status is set when a message is redirected or forwarded.
- Renaming and remove Mailboxes when the Show Subscribed option is enabled now renames/removes the Mailbox and submailboxes from the Subscription list.
- The Composer encodes non-ASCII attachment file names.
- URLs for non-ASCII attachments are now composed in the UTF8 charset (to work with Windows Internet Explorer).
- Address Books records are sorted now.
- Attachments in the AppleDouble format are displayed correctly.
- Auto-wrap algorithms have been changed.
- The autoWrap "flowed" option has been added.
- The charset parameter of message parts containing the ASCII-only symbols is ignored now.
- The directory processing has been changed, the directory.wssp and the sessiondirectory.wssp pages have been modified.
- Now when Sent and Drafts Mailboxes are auto-created, they are auto-subscribed to and the Mailbox list is refreshed.
- More tags and tag parameters are cleaned out from HTML message portions now.

## Directory

- The folderIndex and adminDomainName attributes have been added to the CommuniGateDirectoryDomain objectClass in the default Schema.
- The mailListSettings and groupOptions attributes have been added to the default Directory Schema.

- The fileData attribute and the CommuniGateWebInterface objectClass have been added to the default Directory Schema.
- The CAChain attribute has been added to the default Schema.
- The mailListSettings and groupOptions attributes have been added to the Directory Schema.
- Spaces around the comma signs are removed from the DN string.
- The "search"-type operations now enter the subtrees stored on different Units.

## Foundation

- Thread priority routines are implemented.
- Linux: the STTask routines have been changed to avoid leaving "zombies" of killed processes.
- BSD: the STDictoryEnumerator routines have been changed to avoid directory-detection problems on NFS filesystems.
- AIX: build parameters have been modified to support 6000+ threads.
- Startup: Unix startup scripts have been modified to support a custom Startup.sh file in the {Base} directory.

## LDAP

- Start TLS (RFC2830) is supported now.
- The Compare operation is implemented.
- LDAP-based provisioning for regular domains is implemented (see the Directory Integration section of the manual).

## SNMP

- The WebAdmin Monitor interface to SNMP data is implemented.
- The "total number of jobs"-type parameters now have the Counter data type.
- HTTP monitoring elements have been added.
- Processing of "not-found" elements have been changed to match the SNMPv2 specs.

## ROUTER

- The parser has been changed to allow special symbols in the left part of the alias records (records like `<FAX=*>` = * can be used now).
- Now account alias records can be used for the Central Directory-based Routing.

## RPOP

- Now the Leave On Server option works with any types of UIDs remote servers present and the list of retrieved UIDs is preserved between Server restarts.
- The APOP option is added to individual RPOP records.
- When retrieving mail from Unified Domain-Wide Accounts without using Special Headers, the module now checks that To:/Cc: addresses can be routed to a Local account (rather than just checking that they are directly addressing the Main Domain).

## Cluster

- The "virtual Mailbox" code has been redesigned.
- Mailing List Archives can be browsed via any cluster member.
- The GETLIST and UPDATELIST commands work with any Cluster member.
- The Create/Rename/remove operations for Forwarders and Groups work with any Cluster member.
- Accounts in Shared Directory-based Domains can be administered from any Cluster member.
- The Cluster-Wide Rules are implemented.
- The Cluster-Wide Router Table is implemented.
- Cluster-wide Default Domain Settings, Default Account Settings, Alerts, and WebSkins are implemented.

- The cluster-wide Protection settings are implemented.
- The Dynamic Cluster Controller now tries to move the owner account to itself before creating a mailing list.
- The IMAP and ACAP backend login responses now carry the "Relay" flag.
- Slave startup procedure has been changed to avoid problems on systems with a large number of shared domains.
- Header fields added with the frontend Server Rules are now stored by backends.

## SMTP

- The TURN command is implemented to support dial-up client sites running Microsoft Exchange servers.
- Now the ATRN command is always accepted from non-client addresses (after the AUTH command).
- A workaround for sending mail to buggy firewall relays is implemented.
- The Wake up Now button has been added to the SMTP Settings WebAdmin page.
- The "send encrypted wherever possible" option is implemented.
- When message sending fails because the receiving host drops the connection, the message is re-enqueued (to avoid queue blocking for hosts that violate standards and just drop connections when they do not want to accept certain messages).
- Input messages exceeding the size limit are received, but they are not stored in files.
- The Advertise NTLM AUTH option is implemented.
- Now when the module sends the STARTTLS command, it uses the SSLv3 (rather than SSLv2) "hello" operation.
- The Force AUTH option is implemented.

## Local Delivery

- The `X-Special-Delivery: test` header field now works with the Local Delivery module, too: messages with that field are not stored in the Account Mailboxes.
- If a message is delayed by a Cluster backend, the entire account queue is suspended now.
- The SNMP statistics elements have been implemented.

## POP

- The parameterless AUTH command is supported now (for MS Outlook compatibility).
- Access to empty INBOX Mailboxes has been optimized.
- The STARTTLS/STLS option is seen in the CAPABILITY response only if the addressed Domain has the Security Certificate option enabled.

## IMAP

- The FETCH algorithms have been changed to avoid loading large messages into memory.
- The STARTTLS/STLS option is seen in the CAPABILITY response only if the addressed Domain has the Security Certificate option enabled.
- Non-standard parameters of the Content-Type and Content-Disposition fields are retrieved now.

## CLI

- The Cluster-wide versions of the commands dealing with Default Domain and Default Account settings are implemented.
- The Alert Administration commands are implemented.
- The LISTSUBSCRIBERS command is implemented.
- The WRITELOG command is implemented.
- Banned Header, Protection, and Cluster Protection commands are implemented.
- GETACCOUNTSUBSCRIPTION and SETACCOUNTSUBSCRIPTION commands are implemented.
- GETMAILBOXALIASES and SETMAILBOXALIASES commands are implemented.
- The GETCLUSTERRULES and SETCLUSTERRULES commands are implemented.
- The GETCLUSTERROUTERTABLE and SETCLUSTERROUTERTABLE commands are implemented.

- The GETACCOUNTEFFECTIVESETTINGS and GETDOMAINEFFECTIVESETTINGS commands are implemented.
- The [GET|UPDATE|SET][ACCOUNT|DOMAIN] commands have been renamed into the [GET|UPDATE|SET][ACCOUNT|DOMAIN]SETTINGS commands. Old names continue to work, too.
- The [GET|SET][CLUSTER]DIRECTORYINTEGRATION commands are implemented.
- The Skin Administration commands are implemented.
- The GETCLIENTIPS and GETBLACKLISTEDIPS commands are implemented.
- The GETWEBUSERSESSION and KILLWEBUSERSESSION commands are implemented.
- The SetAccountPassword command is implemented.

## LIST

- when archive Mailboxes are swapped, the newly created archive Mailbox gets the ACLs of the old archive Mailbox.
- The "Hide From Address" option is implemented.
- RFC2919 (LIST-ID header field) is implemented.
- Additional subscriber address checks are implemented to avoid self-subscribing of the special mailing list addresses (-on, -subscribe, -off, etc.)

## PIPE

- Foreign Queue processing is implemented.
- [STDERR], [FILE], [RCPT], and [RETPATH] tags are implemented.

## Rules

- Reply and React texts can start with a [charset] prefix.
- The Vacation Rule priority is not set to 2, to make the server apply it before the Redirect All Rule.

## Directory-Based Domains

- Groups are supported now.
- Custom WebUser and WebAdmin Interfaces are supported now.
- Skin support is implemented.

## Bug Fixes

- Bug Fix: Cluster: the "addMessages" operation did not work correctly with Virtual Mailboxes.
- Bug Fix: CLUSTER: WebAdmin: Personal WebSite administration did not work across Cluster members.
- Bug Fix: ROUTER/DNR changes in 3.5b1 effectively disabled the SMTP RBL feature.
- Bug Fix: RULES: the Reply/React operations in 3.5b1 incorrectly inserted the From:/Sender: header field.
- Bug Fix: WebUser/WebAdmin: in 3.4b3 version custom file uploading could fail in regular (non-directory-based) Domains.
- Bug Fix: CLUSTER: renaming a shared domain did not rename Domain Aliases on all cluster members.
- Bug Fix: WebUser: non-standard MIME content types and subtypes could be returned corrupted.
- Bug Fix: new (Skin-based) List Archive Browser pages did not have the correct charset specified.
- Bug Fix: old (Web-User and WebAdmin) List Subscribers pages could crash the server if the list had the Require Confirmation option disabled.
- Bug Fix: Skins: the default rules.wssp and rule.wssp files did not have the closing </SELECT> tag, causing problems for Netscape browsers.
- Bug Fix: Skins: the webUserSiteIndex internal code routine misplaced the account name (was shown as an error code).
- Bug Fix: WebUser: the Auto-wrap algorithm for "flowed" texts could cause crashes on some platforms.
- Bug Fix: HTTP: the Redirect operation placed "http://" into the Location header even if the current connection was an https one.

- Bug Fix: Viewing the "Queue" Monitor page could cause Queue deadlock.
- Bug Fix: Rules: if the Reply/Reply All operation parameter was specified with the "+" sign and additional headers, the Cc: headers were not processed at all, and Bcc: headers caused parsing errors.
- Bug Fix: Mailbox renaming for non-top level Mailboxes did not work with the "rename Submailboxes" option.
- Bug Fix: Directory: Remote Units: the Search operation returned "unstripped" DNs when the "Server Base" setting was non-empty.
- Bug Fix: 3.3-3.5b9 a rare deadlock situation (in all prior versions) could stop ENQUEUER and DEQUEUER processors if an administrator opened the Message Monitor page.
- Bug Fix: 3.2-3.5b9 message file stored in the MDIR Mailboxes by Cluster backends could be improperly replaced with file links if the Reuse Temp Files option was enabled.
- Bug Fix: 3.4-3.5b9: supplementary send-phase relay checking algorithms were not the same as the input-phase algorithms, causing relay refusals in some rare situations.
- Bug Fix: 3.3-3.5b9: Directory: Local Unit: the delete record operation incorrectly checked if record children existed.
- Bug Fix: 3.4-3.5b9: CLUSTER: the POP3 "relaying enabled" flags were not processed correctly on frontends.

---

**3.4** 28-Feb-01

## Platforms

- OpenBSD/Intel version is released.
- BeOS/Intel version is released.

## Domains, Accounts, Objects

- Directory-based Domains: Personal WebSite support is implemented.
- Directory-based Domains: Account Removal is implemented.
- Directory-based Domains: Account Renaming now renames the Account files, too - if they reside within the DirectoryDomains file directory.
- Directory-based Domains: Forwarders are implemented.
- Directory-based Domains: when account is auto-created on first access, the INBOX Mailbox and other account details (suppl. Mailboxes, Mailbox aliases, subscriptions, etc.) are automatically created, too.
- Statistics on received messages (per Domain) is collected now.
- The global DomainAliases.tdb file is phased out. Now the DomainAliases.data file in the Settings directory of the Domain subdirectories is used to store this domain aliases. The content of DomainAliases.tdb is moved automatically to those files, and the DomainAliases.tdb file is renamed into DomainAliases.tdb.unused.
- Domains: the Send To Forwarders option has been added to the Mail to All panel.
- Template: Initial (default.html) page for Personal WebSites can be specified now.
- Template: Initial (Greeting) message can be specified now.
- Forwarders: addresses without '@' and '%' symbols are qualified using the forwarder Domain name.
- Account: the Collect Account Statistics setting is implemented (the Obscure page).
- Account: the new method is used to recalculate the MailStore size stored in the .info files.
- Passwords: the --BatchLogon command line parameter is documented.
- Passwords: now the U-crpt password encryption can be used on Windows-based servers, too.
- The Consult External Authenticator Domain Settings is implemented.

## Mailboxes

- MDIR Mailboxes are now parsed correctly even if the message internal date was set to 0.
- The ACL subsystem has been redesigned.
- Foreign Mailboxes can be renamed and deleted now.
- The Mailbox view creation algorithm has been changed.
- The "~username" Mailbox aliases are implemented now - they provide access to all shared Mailboxes in the

specified account.

- Now all 'outer' Mailboxes are automatically created when a sub-mailbox is created.

## Queue

- ENQUEUER design is multi-threaded now.
- The Dequeuer messages can be customized now (using the General->Strings page).
- the Copy Failure Reports option is implemented.
- Delivery delay warnings are implemented.

## Clusters

- A separate Domain Controller is implemented.
- The Account Controller is completely redesigned.
- The Account and Domain Controllers now automatically move to a different backend server if the current Controller backend server fails.
- Error codes reported by backends are now transparently relayed via frontends to client mailers.
- POP Alert messages generated on backends are now relayed via frontends to client mailers.
- Backend IMAP alerts issued at the login time are now passed to the client.
- Errors on inter-server SMTP connections do not turn on delays and other anti-attack mechanisms.
- Personal Web Site support now works in both Static and Dynamic Clusters w/o exceptions.
- Domain Aliases are now properly initiated during the frontend and "slave" backend startups, and the domain aliases are properly removed from all servers when the domain is removed.
- Non-local IP Addresses can be assigned to shared domains in Static Clusters.
- Domain Aliases now work for Shared Domains (Dynamic CLUSTER).
- Directory-based Domains can be used for Static Clustering now.
- "Virtual" Mailbox objects are implemented (allowing an Account opened on one server to access Mailboxes in an account opened on a different server).

## Router

- ROUTER: algorithms used for Local Domain Routing are redesigned.

## Security

- The PLAIN authentication method now uses an empty challenge string.
- The Hide Unknown Account Error option is implemented and it is enabled by default. See the Security section for more details.
- The External Authenticator internals are redesigned, its settings are moved to the Helpers page.

## Plugins

- Content Filtering/Anti-Virus API is implemented (see the Rules section).
- McAfee scanning plugin is implemented.

## Directory

- Directory: Remote Units: the Server Subtree setting is implemented.
- Directory: many attributes and objectClasses have been added to the Local Unit built-in Schema.
- Directory: the Import LDIF and Import LMOD functions are implemented in the WebAdmin Interface.
- Directory: the Search Results Limit setting is added to the Local Unit settings.
- Directory: Local: the Enforce Schema setting is implemented (it is now enabled by default).
- DIRECTORY: the "cn=schema" subtree is implemented. The Local Unit schema can be retrieved and extended using the 'cn=schema' record.

## Directory Integration

- The "UID Subtree" setting is implemented.
- The `mail` attribute processing can be fine-tuned now. See the LDAP Module chapter for the details.
- The 'Store Passwords for Regular Accounts' option is implemented.
- The hostServer attribute can be renamed now.
- Forwarder records are now included into the Domain directory subtree.

## POP

- The message size has been added to the RETR command response (to make Netscape mailer show its progress bar correctly).
- The login response line contains the total size of all messages in the Mailbox (to make the Netscape e-mail client happy).

## ACAP

- The PREFIX, SUBSTRING, and SUFFIX filters are implemented.

## HTTP

- The Personal Web Site prefix is now detected after URL decoding.

## IMAP

- The Content-Description field is processed now.
- The MULTIAPPEND extension is implemented.
- RFC2971 (the "ID" extension) is implemented.

## LIST

- The HELP command is implemented.
- The Content-Transfer-encoding header field is removed from distributed 'feed' messages only if its value is '7bit' (previous versions removed the header with the value of '8-bit', too).
- delivery to Lists now works without exceptions in both Dynamic and Static Clusters.
- the "Special" posting mode processing has been changed to make it possible to subscribe lists to other lists.

## RPOP

- Processing is re-scheduled when an RPOP record changes its poll period value.

## PIPE

- The Queue (Wait) page is added to the WebAdmin Monitors.

## SMTP

- Relay To Client settings can be set to "no".
- Relaying settings have been modified.
- The Send Warnings setting is implemented.
- If LDAP connections to the Central Directory fail, the SMTP module now returns a non-fatal error code for unroutable addresses.
- The Local IP Address for outgoing SMTP connections can be specified now.
- The server now reports the "DNS Loop" situation only if a remote host name resolves into an IP Address the SMTP Listener is enabled for.
- The domain name used in the ATRN command must be included into the `Hold Mail for Domains` list.

- The Advertise 8BITMIME option is implemented.

## Local Delivery

- The Send Warnings setting is implemented.
- Delivery to "all" now works without exceptions in both Dynamic and Static Clusters.

## Rules

- Rules: the [RCPT] Execute operation prefix is implemented.
- Rules: The Redirect All simplified Rule now has the "Preserve To/Cc fields" option.

## SNMP

- The CGatePro-MIB.txt file now includes the absolute OIDs.

## WebUser Interface

- The Session Time Limit setting is implemented.
- If the Save a Copy option is selected on the Compose page and the Mailbox with the specified name does not exist, a new Mailbox is automatically created.
- The Thai charsets support is implemented.
- The Reply operation now uses a smarter algorithm to find plain text message portions.
- The Strings.data file now contains the "Translator" dictionary that can be used to translate pop-up menus like those used in the Rule composer.
- The "Print-friendly" message link is implemented, the Strings.data "MessageHeaderEnd" string has been modified.
- The X-UUEncode MIME encoding is supported now.
- The UUencoded files embedded into plain text messages are displayed now.
- The keywords INBOX and Trash can be "translated" in the Strings.data file now, so these "fixed-name" Mailboxes can be "renamed on screen".
- The HeaderNames dictionary is added to the Strings.data file. It can be used to change the names of the RFC822 header fields used on the Mailbox and Message pages.
- Japanese (ISO-2022-JP) characters are now correctly processed in the message header fields on the Compose page.
- The UTF8 mode for the Chinese (Big5 and GB2312) encodings is supported now.
- The UTF8 mode for the Japanese (ISO-2022-JP) encodings is supported now.

## CLI

- The StatReset keys were added to the Domain and Account Statistics data.
- The CREATEWEBUSERSESSION command is implemented.
- The GETWEBFILE, LISTWEBFILES, GETWEBFILESINFO, PUTWEBFILE, DELETEWEBFILE, and RENAMEWEBFILE commands are implemented.
- The GETACCOUNTSTAT, RESETACCOUNTSTAT, GETDOMAINSTAT, and RESETDOMAINSTAT commands are implemented.
- The GETACCOUNTLISTS and GETDOMAINLISTS return value types are changed to *dictionary*.
- The GETDOMAINLISTS is now the preferred name for the LISTLISTS command.
- The MAINDOMAINNAME command is implemented.
- The DELETEMAILBOX, RENAMEMAILBOX commands are implemented.
- The RefreshOSData, GetRouterTable, and SetRouterTable commands are implemented.
- The LISTMAILBOXES, CREATEMAILBOX, GETMAILBOXINFO, GETMAILBOXACL, SETMAILBOXACL, and GETMAILBOXRIGHTS commands are implemented.

## Admin

- WebAdmin: Administrator Preferences are implemented (see the SysAdmin->HTTP section of the manual for more details).
- WebAdmin: the Preferences settings are implemented for POP,IMAP,ACAP,LIST,SMTP,LOCAL,PIPE, and RPOP Monitors.
- WebAdmin: the Preferences settings are implemented for the Domain Account List page.
- WebAdmin: all numeric settings can now be set to some "other" (unlisted) value.
- WebAdmin: now Account Rules can be specified in the Account Import file.
- The "Drop Server Root privilege" options are implemented (Unix only). See the SysAdmin section for the details.

## TLS/SSL

- TLS 3.1 is implemented.
- The Generate Key option is added to the Domain Security page.
- Support for 2048-bit keys and long Certificates is added.
- The Certificate Authority Chain option is implemented.
- The nested TLS negotiations are supported now. They are used to implement strong (128-bit) SSL encryption with weak (40-bit), "export-legal" products.

## Misc

- Migration: the MoveIMAPMail and MovePOPMail utilities now support the --noTimeOut flag.
- The old DataBase Managers (handling the .tdb and .ldb files) are phased out.
- Notifier: the Log Level and Queue size can now be specified using the Obscure page.

## Foundation

- Foundation: the STSkipList data structure is implemented.
- Foundation: the OS Password checking routines now check for the OS Account and Password expiration dates.

## Bug Fixes

- MAILBOX: the 'mailbox list' routine incorrectly capitalized names of 'xxxx/inbox' sub-mailboxes.
- Log recording stopped if the total size of all logs generated since the last server restart exceeded 4GB.
- Domains could not be removed if the domain Default WebUser Preference set has been used at least once.
- WebUser: the default reply text routine could incorrectly insert quotation marks. In some rare cases it could cause a system crash.
- WebUser: not all special symbols in Mailbox names were properly URL-encoded.
- WebUser: if a custom WebUser Interface file was empty, the server could crash.
- TLS sessions interrupted in the negotiation phase could crash the server.
- Domains: when the Main Domain was renamed, the Domain Directory Subtree was not renamed.
- WebAdmin: the size-type parameters could not be set to 'Default (XXX)' values.
- ENQUEUER: Messages with several 'Subject:' headers were processed incorrectly.
- LIST: the 'banned' mode could not be set for lists with disabled archiving.
- ACAP/IMAP: if the AUTHENTICATE command was interrupted, the NO response was returned instead of the BAD response.
- ACAP: data strings with special symbols are returned as literals now.
- POP, IMAP: SASL AUTH methods incorrectly supported the "short-form" syntax.
- Directory: Local Units incorrectly processed some update requests ("add" attribute sets were processed as "replace" sets).
- OS Passwords did not work on the AS/400 platform.
- the default Directory Schema did not contain the "cn" attribute for the "organization" objectClass.
- Directory: Multi-Level searches in Local Units could return incomplete DNs. This could also cause the LDAP module to stop on-the-fly creation of the "mail" attributes.

- Incorrect parallel initialization could cause the PIPE module to crash on startup.
- The RPOP module could accept To: and Cc: addresses as "trusted" when working without the "Special-header" option (this could result in unwanted relays).
- Personal WebSite: URLs for site files did not contain URL escape symbols, some of the access utilities did not remove the URL escape symbols.
- CLUSTER: POP logins could fail on backends.

---

**3.3** 16-Jul-00

**Update Note**: the 3.3 version uses a completely new Directory Manager. If some of your Domains had the Directory Integration setting set to Keep In Sync, open those Domain Settings in the newly installed 3.3 version, and click the "Insert All" button in the Directory Integration Panel.

## Platforms

- OS/400 version is released.
- Linux/Sparc version is released.
- Linux/StrongARM version is released.

## Domains, Accounts, Objects

- Groups are implemented. GROUP-related CLI commands are implemented.
- Forwarders are implemented. FORWARDER-related CLI commands are implemented.
- The Directory-based Domains are implemented.
- The "Mobile" setting is enforced now: domains and account w/o that Access Mode cannot be accessed from non-client IP addresses.
- The 2-Letter 2-Level Domain Hashing now provides a workaround for accounts with 1-letter names and for accounts that have the dot symbol as the second symbol of their names.
- The Generate Index option is implemented. It can be used to decrease the restart time for domains with 100,000+ accounts.
- OS Names can be explicitly set for individual Accounts.
- Automatic "account is over quota" alerts are implemented. See the SysAdmin->Alerts section of the manual.

## Mailboxes

- Mailbox Aliases are implemented (see the Objects->Mailboxes section of the manual). Mailbox aliases can be used to provide access to foreign Mailboxes for IMAP clients (such as MS Outlook / OE) that cannot process foreign Mailbox names in the Mailbox Subscription lists.
- Mailbox aliases are transparent now (included into Mailbox hierarchy views).
- The BSD Mailbox Manager now checks the size of text lines in new messages.
- The .mdir (MailDir) format is redesigned to store the number of message text lines in message file names. This should help some mailers (such as Netscape) correctly process messages with attachments retrieved from Unix servers.

## Queue

- The DEQUEUER engine is redesigned and converted into a multi-threaded one.
- Messages with extremely long header fields (>100K) are now rejected with the SMTP, RPOP, PIPE, or POP (XTND XMIT) modules.

## Router

- The Add Name to Non-Qualified Domain Names option is implemented.

## Directory

- The Directory Manager is implemented.
- The File-based Directories are implemented.
- The LDAP-based Directories are implemented (with caching).
- The .tdb and .ldb DataBase Managers are being phased out.
- Multiple Sub-tree storage units are implemented.
- Schema editor is implemented.
- Remove and Relocate Storage Unit operations are implemented.
- Access Rights (ACLs) are implemented.
- Browser and Access Right WebAdmin pages are implemented.
- File(Local Units): the updates merging daemon is implemented.
- Storage Unit deletion is implemented.

## Directory Integration

- The Directory Integration Settings are implemented (Domains section).
- The special-case `dc` RDN attribute is supported now.
- The Delete All operation now removes only the records for accounts created on this Server.
- Mailing lists and Groups are now automatically added/updated in the Directory if the Domain Integration setting is set to Keep In Sync.

## LDAP

- Implementation is based on the Directory Manager now.
- RFC2254 is implemented.
- The 'mail' attribute is now composed on-the-fly for records of the CommuniGateAccount objectClass.
- The authentication methods are improved and documented.
- Case-insensitive dictionaries are now used in modify-type operations.

## ACAP

- The output buffering method has been changed.
- Datasets entry names are case-insensitive now.

## HTTP

- WebUser port handling has been changed to support domain-style Personal Web Sites (i.e. http://user.domain.com can be processed as http://domain.com/~user/ now).
- CGI environment variables HTTP_AUTHORIZATION and HTTP_REFERER are added.
- CGI program name in a URL can now be followed by '/' and some URL string.
- CGI programs now inherit the environment variables of the Server (under Windows, this is needed to open TCP/IP sockets in CGIs).

## IMAP

- The output buffering method has been changed.
- The APPEND command now checks if the message text lacks the trailing EOL and fixes it. This is a workaround for the Netscape Messenger bug.

## LIST

- The Special posting mode is implemented to allow subscribing of a mailing list to some other list.
- The First Digest At setting processing has been changed (see the LIST module manual).
- The Digest generator has been modified to fit the RFC1153 requirements.

- Feed headers and trailers are correctly inserted into base64-encoded messages.

## RPOP

- Added support for remote POP servers that do not return the number of messages in the initial UIDL response.

## SMTP

- The SIZE EHLO response does not include the '=' sign now.
- The AUTH=address Mail From parameter is now supported.
- Batch-splitting for messages with multiple recipients in one domain is implemented.
- Log reporting of relayed messages is improved.
- The Send Encrypted option is implemented to support server<->server encrypted message transfer.
- RFC2645 (ATRN) is implemented in both server and client modes to support mail delivery to hosts with dynamic IP addresses.
- Secure connections with forwarding servers are supported now.

## Local Delivery

- All-domain aliases are case-insensitive now.
- The new Alert Text option allows you to specify the "over the quota" alert message text.
- The Account Detail addressing Routing options are implemented.

## Rules

- The Current Date and Time of Day Rule conditions are implemented.
- The Write To Log action is implemented.
- Comparison operations now ignore the surrounding angle brackets in the envelope (Recipient, Return-Path) addresses.
- The Current Day conditions are implemented.
- The "[FILE]" and "[STDERR]" Execute command tags are implemented.

## SNMP

- SNMP agent is implemented.
- StalkerMIB ObjectID has been changed to 5678 - the IANA-registered Stalker ObjectID.
- WebUser monitoring agents are implemented.
- SMTP monitoring agents are implemented.
- IMAP monitoring agents are implemented.
- TLS monitoring agents are implemented.
- POP module totaling-type reporting is implemented.
- 64-bit Counters are implemented, some MIB elements have been switched to the COUNTER format.

## WebUser Interface

- Korean, Chinese, and UTF-8 encodings are supported now.
- The Mailbox page now remembers the filter and search strings and the position in the selected message set.
- The Empty Trash button is now implemented on the Mailboxes Page.
- UTF8 Mode Setting is implemented. The Settings.html and Compose.html pages have been modified.
- The Settings.html, RPOP.html pages have been changed (^m has been added).
- Compose: now the From, To, Cc, and Reply-To original letter headers are MIME-decoded first.
- The Use MIME for Headers option is implemented. When this option is selected, the Subject, To, Cc, and Bcc header fields containing non-ASCII symbols are MIME-encoded.
- Alerts, Bye, Hello, List, ListArchive, Public, Rules, Rule, RPOP, Subscribers, Subscription pages now include the '=' (charset) macro symbols.

- the '^$' macro (domain name) now works in all Account pages.
- All ISO-8859-x and windows-125x charsets are now supported in the UTF-8 mode.
- Longer, alpha-numeric "session passwords" are used now.
- The DirectoryFields arrays are added to the Strings.data file.
- The `Files` realm is implemented to provide access to arbitrary files in the WebUser directory.
- Attachment file names and HTML text portions are MIME-decoded and (optionally) converted to UTF-8 now.
- Texts using the Japanese ISO-2022-jp charsets should be displayed correctly now.
- The Directory Search page can now use the "internal CGatePro" names instead of the standard attribute names (i.e. `RealName` instead of `cn`).
- Security: the Security Certificate (RFC2585) link is added to the Login page. See the Security section of the manual.
- Non-ASCII Mailbox names are supported now.
- format=flowed processing (RFC2646) is implemented.
- Korean (ISO-2022-KR) and Chinese (Big5) letters are correctly converted into HTML code.
- The Certificate link is displayed only if the domain has a Custom Certificate.
- The Mailing Lists link is displayed only if the domain has some mailing lists.
- The Sent and Draft Mailbox names are properly "defaulted" now. The WebUser Settings.html page has been changed.
- The simplified Chinese (GB2312) charset is supported now.

## Admin

- The WebSite, BasicAccountSettings, and WebUserSettings Domain Access Rights are implemented.
- The --noLockFile command line option is implemented.
- Clicking the Refresh button on the General Settings page tells the Server to re-read the DNS addresses from the OS.
- The Account Template page now includes the Initial Mailbox Aliases panel.
- The Obscure WebAdmin Page: the Central Directory settings are phased out.
- The GetAccountLocation, GetServerRules, and SetServerRules CLI commands are implemented.

## TLS/SSL

- client-side secure connections are implemented.
- the SSL 3.0/3.1 interaction is improved (interoperability with both Microsoft products and open source utilities).
- TLS session recycler is implemented.

## Foundation

- DNS addresses are now correctly retrieved from the Windows 2000 Registry.
- A faster version of 'write to file' method is implemented (now used to store aliases, forwarders, groups, and account info).

## Manual

- The WebMail section is created to contain all user-level information about the WebUser Interface.
- The HTTP and HowTo sections are updated to provide the information about username.domain.dom Personal Web Site URLs.
- The Directory, Directory Integration and Clusters pages are updated.

## Bug Fixes

- CLUSTER: slave nodes refused to connect to the controller if one of the shared domains had the WebCache option disabled.
- SMTP, RPOP: if a remote server was misconfigured, garbage left in the response buffers could cause problems when connecting to other servers serving that remote domain.

- the Header-field Rule condition did not process wildcard symbols correctly.
- WebUser: the "New" counters on the Mailboxes page were not updated properly.
- SMTP: the "relayHost" field was not always filled correctly resulting in garbage on the SMTP Monitoring pages.
- CLI: GetAccountAliases command could crash the server is the specified domain did not exist.
- ACAP: multi-level searches in Dictionary DataSets could cause synchronization deadlocks.
- ACAP: MODTIME responses for the SEARCH command were returned untagged.
- ACAP: the optional metadata list in the RETURN clause of the SEARCH command was not properly parsed.
- under IRIX, AIX, HP/UX daylight saving times for the local time zones was not detected correctly.
- Rules: Rule Editor could crash the server if some of the condition or action fields were missing in the (customized) form.
- POP: the CAPA command was not processed in the TRANSACTION state.
- Binary zeros in message headers could crash the server.

# Version 2.x Revision History

- 6.2
- 6.1
- 6.0
- 5.4
- 5.3
- 5.2
- 5.1
- 5.0
- 4.3
- 4.2
- 4.1
- 4.0
- 3.x
- 2.x
- 1.x

RSS **2.9.1** 09-Mar-99

- Bug Fix: the new Router scanner did not cut trailing spaces from the parsed names.

**2.9** 08-Mar-99

- PIPE: delivery to external applications is implemented.
- WebUser/Domains: The Add Trailer Domain Option is implemented.
- Router: the parser is modified to accept quotation marks in addresses.
- WebUser: the In-Reply-To header is no added to reply messages composed using the WebUser Interface.
- IMAP: part filename and charset data are now included into the BODYSTRUCTURE responses.
- LIST: now existing subscribers can always confirm their subscriptions and can unsubscribe.
- Bug Fix: the All Domains And Account Settings access right worked as Master Right in 2.9b3-2.9b5 versions.
- Bug Fix: Redirect/Forward/Mirror operations in the Account-Level Rules used the empty (<>) address instead of the full account name.

**2.9b5** 27-Feb-99

- WebUser: personal account Web Sites are implemented. Users can build their Web sites using any HTML editor that uses the PUT/DELETE/MOVE HTTP methods to upload pages (Netscape Composer and others).
- Admin: the MaxWebSiteSize and MaxWebSiteFiles settings (limiting the personal Web site size) are added to the Account Settings.
- The Central Directory can now contain any number of fields. These fields can be set in Account Settings and in the Account Templates.
- The `maildir` Mailbox type is implemented. See the Account -> Data section of the manual.
- WebAdmin: Server and Domain administrators can now retrieve and update the WebUser Interface files via

HTTP using the products like the Netscape Composer.

- WebUser: when displaying a message, the charset of the HTML page is set to the first non-ascii charset used in the message (making it possible to view non-roman letters w/o manually selecting the charset in the browser).
- WebUser: if the Display Subscribed option is enabled, the account subscription is updated when Mailboxes are renamed and/or removed via the WebUser interface.
- Rules: the `Each To and CC` condition is implemented.
- LOCAL: Unified Domain-Wide Accounts can be created in Secondary Domains.
- SMTP: the module does not try to resolve non-qualified HELO/EHLO names now.
- Security: the "Plain" method has been updated to support all versions of Netscape Messenger.
- Bug Fix: if an HTTP connection broke while the Server was receiving a non-form POST request, an exception was raised.
- Bug Fix: Auto-Replying and Reacting to messages w/o the Subject field crashed the server.
- Bug Fix: the LDAP server did not interpret zero time-out value as "unlimited".
- Bug Fix: the LDAP server crashed when generated time-out reports.
- Bug Fix: 2.9b3-4 versions interpreted the "Assign IP Addresses by MX record" setting as the "By A-Record" setting.

**2.9b4** 18-Feb-99

- The first BSDI BSD/OS version is released.
- SMTP: a new option tells the SMTP module not to advertise the SMTP AUTHentication feature (to avoid problems with Netscape 4.x Messenger).
- SMTP: Verify Return-Path processing has been changed.
- POP: Access to Individual Mail in Unified Accounts is implemented.
- CLI: the GetModule, SetModule, GetDomain, and UpdateDomain commands are implemented.
- HTTP: request `Content-Length:` values are validated now.
- WebAdmin: all size settings can be set precisely now, using the `Others` menu option.
- WebUser: the Strings.data dictionary file is added to keep customizable HTML elements used in WebUser Interface.
- WebUser: the Public Info editor is implemented.
- Domain Admin: the Per-Domain Accounts and Lists Number limits are implemented.
- Rules: processing of address strings (conditions) has been changed.
- Rules: Web editing has changed to avoid problems with mixed links.
- The `--NoExceptions` option is implemented (was set by default in all 2.8-2.9 versions).
- Bug Fix: POST HTTP requests w/o the multipart encoding resulted in memory leaks.
- Bug Fix: a Server-Wide Rule could crash a server if it discarded or rejected a message.

**2.9b3** 05-Feb-99

- The "Execute" Rule action (allows users to start external programs) is implemented.
- The DNR listener now logs all error conditions instead or shutting down the server (these situations can be met rather often on Linux systems).
- The SMTP module now detects a single '%' sign in an E-mail address and changes it into the '@' sign before sending to a remote host (see the Router section of the SMTP module guide).
- The SMTP module can now accept remote queue starting commands for the Client Hosts only.
- RPOP: Special Headers are removed from the received messages.
- Bug Fix: not all local IP address were detected on some Unix platforms.
- Bug Fix: if a Content-type or Content-Disposition message header had an incorrect format, the MIME parsing engine could crash.

**2.9b2** 22-Jan-99

- The ACAP module is implemented.
- The DataSet manager (used with ACAP) is implemented.

The Server-Wide Rules are implemented.

- The SASL Authentication methods (RFC2222) are implemented. See the Security section.
- SMTP: SASL authentication is implemented; this can be used to allow relaying for mobile users and to submit "trusted" messages. The later can be used for LIST approval via E-mail and other operations.
- WebAdmin: the Domain-level administration is implemented. See the SysAdmin section.
- WebUser: the IP-address controlling mechanism can be disabled - useful for users accessing the server via a multi-homed proxy.
- WebUser: if the Display Subscribed option is enabled, the account is automatically subscribed to all new Mailboxes created via the WebUser interface.
- WebUser: the width of the Message field can be specified in the User settings now.
- IMAP: the "CREATE Mailbox/" operation now creates all intermediate folders (needed to support the Eudora mailer in the IMAP mode).
- RPOP: the polling scheduler internal design has been modified.

**2.9b1** 20-Dec-98

- ITU "BER" decoder/encoder is implemented.
- LDAP server is implemented (provides read-only access to the "Central.tdb" database).
- SMTP, POP, IMAP, PWD, and LDAP port numbers can be specified now.
- SubFolder support for large domains is implemented.
- the `AllDomains` virtual account/address is implemented. A message sent to that address in the main domain is stored in all accounts in all server domains.
- Bug Fix: if a message was rerouted/forwarded to an empty set of valid addresses, the operation was not rejected; instead, an incorrectly formed message was submitted and then moved to the Bad Files.

---

**2.8** 06-Dec-98

- Bug Fix: the long-standing bug that first showed up on MacOS and recently reported on some Linux and Solaris systems was finally found and fixed. The bug affected all Unix-based systems, it appeared in rather specific situations, and it could (later) cause various problems - the server stopped to respond, Web clients saw the "Status :1" browser errors, queued message files were stored as empty files, etc.
- The first Digital Unix (OSF) version for Alpha® processors.
- POP: the CAPA operation (RFC 2449) is implemented.
- Mailbox Management: a "quick check" is now performed before a message is stored into an unparsed Mailbox. These decreases the probability of storing messages with duplicate UIDs, especially when working with `external Mailboxes`.
- LIST: NULL and BANNED subscription types are implemented.
- LIST: the Archive Swap option is implemented.
- Bug Fix: LIST: the `first digest` and `cleanup period` options were not restored correctly after a restart; this could cause digests not being generated in time.

**2.8b3** 29-Nov-98

- CLI: the administrator command line interface (CLI) is implemented (see the API/CLI section of the Guide).
- Account/Mailboxes: External Mailboxes providing legacy Unix mailer compatibility are implemented.
- WebUser: an empty WebUser folder is now created inside Domain folders. HTML and other files placed into that folder override the files in the main WebUser folder. This allows the system administrator to create different WebUser interfaces for different domains.
- The ProcessID lock-file is implemented (stored in the base directory). Unix start-up scripts now can use that file to stop the Server.
- The --NoCatch, --LogAll, and --Daemon command line options are implemented.
- SMTP: the leading @ sign is ignored in the domain names specified with the ETRN command (needed to serve Lotus Notes client systems).
- Domain service files (settings, templates, rpop accounts, and aliases) have been moved into the special Settings

subfolder. This version of the server should move your files into the new locations automatically.
- Bug Fix: distributing mail to all Domain Accounts via the all@domainname address was broken.

**2.8b2** 17-Nov-98

- Domains: the Assigned IP Addresses option is implemented.
- Kernel: the TCP Activity Scheduler is implemented (restricts SMTP sending and RPOP polling activity).
- Access: the Grant Access to Client Hosts Only option is implemented.
- LIST: bounce processing has been improved.
- LIST: the `listserver` address for List Server requests is supported now.
- POP: access to all Account Mailboxes is implemented.
- POP: access to public and shared (foreign) Mailboxes is implemented.
- HTTP: image files are retrieved with 24 hours expiration period to avoid unnecessary requests.
- Bug Fix: the HTTP module could duplicate empty lines in form data.

**2.8b1** 08-Nov-98

- Access: the local domain resolver uses MX records now (see the Access section of the Guide).
- Access: the "Connections to unassigned IP addresses" option is implemented; it simplifies setup for single-domain systems.
- WebUser: both Frames and No Frames interfaces are implemented (user-selectable).
- WebUser: Web pages are cached now; if you change WebUser pages and want the new pages to be used immediately, start the server with the `--NoWebCache` option.
- LIST: the Enable Archiving & Digesting option is implemented.
- Bug Fix: the LIST module could crash if a message did not have a [valid] From address.
- Bug Fix: the INBOX Mailbox was not visible in single-mailbox Accounts.
- Bug Fix: "Authenticated Users Become Clients" processing is fixed and improved.

---

**2.7** 26-Oct-98

- Kernel: access to foreign (shared) Mailboxes is implemented.
- IMAP: operations with foreign Mailboxes are implemented.
- IMAP: RFC2086, ACL (access control lists) are implemented.
- IMAP: RFC2342, NAMESPACE request command is implemented.
- IMAP: RFC2359, UIDPLUS protocol extension is implemented.
- WebUser: Access to foreign (shared) Mailboxes is implemented.
- WebUser: Access to subscribed Mailboxes is implemented.
- WebUser: Modification of the subscribed Mailbox list is implemented.
- WebUser: Mailbox management (Mailbox renaming/removing and ACL management) is implemented.
- LIST: the Silent, Send Welcome, and Ask Confirmation subscribe operations are implemented.
- LIST: The digest size limit can be set to zero to force digest distribution in the semi-feed mode.
- LIST: max archive size can be set to zero to disable list archiving.
- LIST: the confirmation ID is generated only once now, (not for each new request). The old scheme created too many problems when several warning/confirmation messages were sent.
- LIST: "text alternative" as available as an allowed format (a messages should be either a text, or multipart/alternative with the first alternative being a text).
- LIST: a bug that prevented archive clean-up has been fixed.
- RULES: the COPY command can use the foreign Mailbox (~username/mailbox) name.
- RULE: the Reject Action is implemented.
- RULE: the Add Header action is implemented.
- Secondary Domain Access improved, the Unknown Network Address error should not appear in most situations.

**2.7b2** 20-Oct-98

- Auto Sign-up Domain option is implemented.
- WebUser: Auto Sign-up interface is implemented.
- IMAP: the \Deleted flag is processed in accordance with IMAP4rev1 standard now.
- IMAP: the FETCH RFC822.HEADER.LINES and RFC822.HEADER.Lines.NOT commands are implemented to support old IMAP4 clients.
- IMAP: RFC2087 (QUOTA extension) is implemented.
- IMAP: RFC2221 (Login Referrals) is implemented.
- LIST: digest separator is shorten to 70 symbols so it will not create problems for some mail clients.
- Router: IP address to local domain name conversion has been moved from the SMTP router to the kernel router, so it can be applied when routing addresses for Access operations.
- ACCOUNT: the Can Modify Password option is implemented. The PWD module checks this option now.
- WebUser: password modification is implemented.
- WebUser: the maximum and used account storage is displayed.
- WebUser: the "From Address" setting is implemented.
- SMTP: the "white hole" processing is implemented.

**2.7b1** 16-Oct-98

- Account aliases are implemented.
- Account routing (see the Access section of the manual) is implemented.
- A lot of internal changes and code clean-ups. Recovery procedures. 64-bit-clean code.
- The Return Failed option is implemented (in the Obscure settings).
- The Return-Path and MessageID Rule conditions are implemented.
- Bug Fix: messages rejected with non-fatal error code could be suspended for a huge period of time.

---

**2.6.4** 07-Oct-98

- The socket "send" call processing has been changed. This should eliminate some problems with FreeBSD and slow links.

**2.6.3** 01-Oct-98

- The statically-linked (.tgz) Linux/Intel version for old and non-RedHat systems.
- DNR is improved and fixed (the DNR timeouter thread is retired).
- The resolv.conf nameserver address 0.0.0.0 is not rejected now.
- Several fixes/improvements in Linux and Solaris installation procedures.
- Bug Fix: Thread implementation on Mach fixed (this bug could crash MacOS X Servers).
- Bug Fix: Rerouted addresses processing fixed (this could treat redirected mail as illegal relaying).
- Bug fix: the List Manager could crash when unsubscribing users via Web.
- Bug fix: on several platforms List Manager could enter a deadlock state when subscribing users via Web.

**2.6.1** 19-Sep-98

- The packaged (.rpm) Linux/Intel version.
- The "To", "Cc" and "To or Cc" rule conditions are implemented.
- Bug Fix: some message/digest letters were parsed incorrectly.

**2.6** 03-Sep-98

- **The first commercial release**
- The first Solaris/Intel version ("pkg_add" format).
- The packaged FreeBSD/Intel version.
- Socket diagnostics are improved on all platforms.
- IMAP Mailbox name processing changed to ignore leading "/" symbols (required for the Solaris mailer and

some other mailers).

- Bug Fix: the LIST module did not store the number of processors in the settings file.

**2.6b4** 31-Aug-98

- Account cache is implemented.
- Account hashing tables are implemented.
- Mac OS X (aka Rhapsody) version is now one Installer.app package with fat (PPC + Intel) applications.
- User Web access: the specified domain name is processed with Router domain records.
- Account access: account names are processed with the Router table.
- Local Delivery: the wildcard ('*') character in supported in the Reroute-to Domain Options setting.
- IMAP: the List command is made case-insensitive on servers using case-insensitive file systems.
- RULES: string comparisons are made case-insensitive.
- Some default records are stored in the Router table when the system is installed for the first time.
- Bug Fix: IMAP SEARCH BODY bug introduced in 2.6b2 is fixed.
- Bug Fix: LIST module could crash if a new subscriber had no 'real name' in the E-mail address.
- Bug Fix: Web Admin: Router Settings were not updated properly on Unix systems.

**2.6b3** 25-Aug-98

- RULES: Messages generated with Redirect, Forward and Mirror operations now include the X-Autogenerated header.
- LIST: distribution messages are sent with the Precedence: list RFC field.
- LIST: distribution messages are sent with the Sender RFC field.
- LIST: subscriber address search is case-insensitive now.
- LIST: X-ListServer: field is added to warning, confirmation and other service messages.
- FOUNDATION: implemented a workaround for the Rhapsody/MacOS X bug, so all local IP addresses are retrieved correctly now.
- Bug Fix: the APPEND IMAP command could crash on Win32 platforms.
- Bug Fix: if a message was in the non-multipart text/html format, the User Web interface did not display it correctly.
- Bug Fix: RULES: the Mirror-To operation was not recognized.

**2.6b2** 20-Aug-98

- LIST module: Web interface to mailing lists is implemented.
- LIST module: Web Interface to subscription lists is implemented.
- Search operations for Mailboxes and mailing lists made available via Web Interface.
- DNR is changed to use variable-length time-outs.
- DNR is closed earlier when the server shuts down to abort pending DNR requests.
- Stream management is changed to provide quicker and reliable server shutdown.
- Account caching is implemented.
- QUEUE (for queued messages) Log records are separated from the SERVER Log records.

**2.6b1** 10-Aug-98

- The LIST module is implemented.
- The Listener module changed to make listening sockets stay alive even after network errors.
- DNR datagram socket buffer has been increased to avoid DNS packet loss under a heavy load (when sending mail to large mailing lists).
- Web Viewer: now "multipart/digest" messages are displayed correctly, as a set of RFC822 messages.
- the Mirror To operation is added to the Automatic Rules.

**2.5b3** 31-Jul-98

- Vacation Message processing is implemented now.
- The LOCAL Delivery module is multi-threaded now.
- The LOCAL Delivery module monitor is implemented.
- The Original Recipient (ORCPT) option is supported in the SMTP module and Dequeuer.
- The HTTP "Host" header is now used for multi-domain Web access.
- The "X-Listserver", "Precedence: bulk" and other special headers are now detected.
- The ""Human Source" condition has been added to the Rules.
- Bug Fix: Transferring messages between Mailboxes using Web Interface could crash.
- Bug Fix: On Unix systems the Web Interface did not process long data (message bodies, etc.) correctly (the last symbol was multiplied sometimes).
- Bug Fix: The POST forms were stored corrupted, thus the first parameters could be lost (was seen as incorrectly processed From/Cc addresses in the Web Interface Composer).

**2.5b2** 23-Jul-98

- Bug Fix: Rule Editor could crash when editing Action parameters.
- Bug Fix: Message Viewer did not set the proper content-type for non-text and non-image message components (this could corrupt attachments when they were downloaded via the Web Mail interface).

**2.5b1** 22-Jul-98

- Web-based Mail Access is implemented.
- Automated Mail Processing (Rules) is implemented.
- HTTP "post" requests are supported and used now.
- HTTP MIME-mode parameters are supported and used now.
- Message Redirection/Forwarding is implemented.
- Rule Editor is implemented.
- Bug Fix: SMTP module did not reschedule a message if connection dropped after one of recipients had been rejected.
- Bug Fix: IMAP "LITERAL+" processing is fixed.
- Bug Fix: MAIL-DEAMON is renamed into MAILER-DEAMON and messages to that address are discarded as those sent to the NULL address.

**2.2b1** 20-Jun-98

- Many internal changes in queue processing and error reporting.
- The SMTP Module, RPOP Module, Host monitor and message Monitors are implemented.
- The DNR channels number restriction is implemented (can be used on slow links).
- "White-Hole" processing is implemented, error reporting for blacklisted addresses is improved.
- The "Relay to backed-up hosts" option is implemented.
- The "*-wakeup" and "*.smtp" special addresses are implemented.
- The "Authenticated Users become Clients" option is implemented.
- Settings updates are recorded in the Log now.
- SMTP host delays are restored after a server restart.

**2.1b3** 10-Jun-98

- The first FreeBSD® version.
- The Text Mailbox format has been changed to make it compatible with legacy "mail" programs.
- The PIPE module is implemented.
- The "Use System Password" option is implemented on Unix platforms (employing the "passwd" authentication).
- The Account Templates are implemented.

**2.1b2** 27-May-98

- The first Solaris® version.
- The "Use System Password" option is implemented for the WindowsNT platform (using the LogonUser call).
- The MS Windows Installer is implemented.
- The CGStarter.exe is implemented to enable the CommuniGate Pro Server to run as a "service" under WindowsNT.
- Bug Fix: the POP/IMAP modules could crash during multi-access sessions if one of the sessions has emptied the Mailbox.

**2.1b1** 25-May-98

- The RPOP module is implemented (retrieving mail from external accounts via POP protocol).
- The XTND XMIT extension is implemented in the POP module. It can be used with Eudora to submit mail via POP connections.
- Bug Fix: the Domain options "Mail to Unknown" and "Mail to All" did not work properly on all Unix platforms.

**2.0b3** 11-May-98

- The first Linux version.
- The first Rhapsody DR2 version.
- Rhapsody: the default location for the Server software changed to /Local/Servers, the default location of the "base folder" is changed to /Local/CommuniGate. If you used the older versions under Rhapsody, move the "base folder" and delete the old Server software folder from the /System folder.
- Processing of the IMAP \Deleted flag is changed to confirm the IMAP standard. Non-standard, advanced processing is disabled, and it will be made available as an option later.
- Under Rhapsody DR2, the nameserver addresses are retrieved from the NetInfo database if the /etc/resolv.conf file does not exist.
- On all Unix platforms, the panic (STLog) messages are recorded in the OS "mail" syslog.
- The Web interface HTML pages are updated.
- Bug Fix: the HTTP processor could crash if the very first request resulted in an empty response.
- Bug Fix: deleting of folder accounts, domains and Mailbox subfolders did not work in 2.0b2.
- Bug Fix: the "domain does not exist" DNR error was not processed correctly on all Unix platforms.

**2.0b2** 05-May-98

- Minor fixes in the Web interface. The first Rhapsody version.

**2.0b1** 27-Apr-98

- **The CommuniGate Pro software is rewritten from Objective C to C++.**
- **The CommuniGate Pro software is not based on Apple/Next Foundation framework (Yellow Box) any more, the Stalker Portable Foundation Framework is used instead.**
- Several minor changes in the IMAP protocol.
- Improved performance for extra-large (50MB and more) text Mailboxes.
- Socket library improvements for MS Windows platforms.

CommuniGate **Pro**

| Introduction | Features | **History** | How To | Help Me |

# Version 1.x Revision History

- **6.2**
- **6.1**
- **6.0**
- **5.4**
- **5.3**
- **5.2**
- **5.1**
- **5.0**
- **4.3**
- **4.2**
- **4.1**
- **4.0**
- **3.x**
- **2.x**
- 1.x

RSS

**1.0b4** 08-Mar-98

- Multiple Account Domains are implemented.
- Hierarchical Mailboxes are implemented.
- RFC1870 - SMTP "SIZE" extension support is implemented.
- RFC2180 - IMAP multi-access details are corrected.
- RFC2088 - IMAP non-synchronizing literals are implemented.
- RFC1077 - IMAP IDLE command is implemented.
- The Account Info databases are implemented.
- The Last Login Time, Last Login IP is stored in the Account Info.
- The IMAP UIDValidity / UIDNext functionality is implemented.
- The IMAP RECENT functionality is implemented.
- The POP LAST command is implemented.
- Independent UIDs for every Mailbox are implemented.
- Sizes of folder-type accounts are calculated dynamically now.
- The marked/unmarked Mailboxes status is indicated in IMAP now.
- IP multihoming is supported.
- Multi-domain support based on multihoming is implemented.
- Multi-domain support based on @ and % symbols is implemented in POP, IMAP, PWD and HTTP modules.
- Bug fix: when the SMTP module was receiving a letter with a line starting with "." followed with an empty line, the Server crashed.

**1.0b3** 20-Feb-98

- The Rhapsody/Intel version is released.
- Now the SMTP module can open several channels when sending messages to one host.
- Monitor access privileges are added. Logs and Queues panels require the Can Monitor privilege now.

- Queue monitor panel is implemented. It shows all the messages in the Server queue.
- Now the DNR module can repeat requests to Domain Name Servers.
- Now the DNR module can use several DNS servers.
- The POP and IMAP modules now remember the IP addresses used to make authenticated connections. The SMTP module considers those IP addresses as "Client Hosts" for 30 seconds.
- Bug Fix: the DSN messages did not have the "trusted source" marker.

**1.0b2** 09-Feb-98

- Various low-level issues corrected to fix the Yellow Box for Windows version.
- Bug Fix: the IMAP `BODY[]` commands (used with MS Exchange) were not processed correctly.
- Bug Fix: IMAP `SEARCH` date did not work correctly with MS Outlook clients.
- Bug Fix: IMAP module did not correctly processed `UID`-prefixed commands causing problem for some mail clients (including one from Netscape).
- Bug Fix: storing messages in an open Mailbox that had been cleared caused a crash.

**1.0b1** 03-Feb-98

- The SEARCH BODY/TEXT IMAP command is implemented.
- Now non-EXPUNGE IMAP Mailbox updating information can be returned in response to FETCH, STORE and SEARCH commands.
- Bug Fix: IMAP EXPUNGE responses could be returned for the STORE command.
- Bug Fix: The SEARCH KEYWORD IMAP command was not implemented correctly.
- Bug Fix: Account Enabled and Login Enabled options were not set correctly via the HTTP interface.

**1.0b0** 02-Feb-98

- The first public release.

---

| Introduction | Features | History | How To | Help Me |
|---|---|---|---|---|

# How To

- **Routing**
  - **How can I gradually migrate accounts from my old server?**
  - **How can I relay E-mail and Signals for certain domains?**
  - **How can I send E-mail and Signals to a remote host bypassing its DNS MX/SRV records?**
- **SMTP**
  - **How can I forward mail to the other SMTP MTA on the same server?**
  - **How can my customer servers receive mail if they have dial-up connections? (ETRN)**
  - **How can I hold all client mail till their servers send ETRN?**
  - **How can my customer servers receive mail if they have dynamic IP addresses? (ATRN/RPOP)**
  - **How can my customers release mail to all their domains with one ETRN or ATRN?**
- **Rules**
  - **How can I store all outgoing mail sent by all my users?**
  - **How can I restrict to whom my users can send mail?**
- **Mailboxes**
  - **How can I create and use Shared Mailboxes?**
  - **How can an Administrator clean User Mailboxes?**
- **Account File Storage**
  - **How can I provide `username.domain.com` personal Web sites?**

This section explains how you should configure your CommuniGate Pro Server if you have some specific needs..

# Routing

## How can I gradually migrate accounts from my old server?

In many cases, especially when you migrate users from an old server, you may want CommuniGate Pro to deliver mail to all Accounts created in a certain Domain, while mail to all Accounts that do not [yet] exist in that CommuniGate Pro Domain should be relayed to some other [old] server, without any change in the headers and envelope addresses.

Open the Domain Settings for that Domain and set the Mail to Unknown option:

| Unknown Names | |
|---|---|
| Mail to Unknown | Rerouted to |

Here `domain.dom` is the name of this CommuniGate Pro Domain and `otherserver.dom` is the DNS name of the other [old] server. If the DNS name for the other server does not exist, you can use the IP address instead:
`*%domain.dom@[11.22.33.44]`

When the CommuniGate Pro server receives any message directed to `aname@domain.dom`, and the Domain does not have an Account/Group/Forwarder/Mailing List with that `aname` name, the message is Rerouted (the envelope address is changed) to `aname%domain.dom@otherserver.dom._via`. The `._via` suffix tell the Router module to accept this address, and to cut off the domain name, using that part only as a name of the server to connect to (the Router module always cuts off the IP-address domain parts, too). The resulting envelope address (`aname%domain.dom`) is converted to the standard form (`aname@domain.dom`) before it is sent to that other server. As a result, the other server receives such a message with the unmodified envelope data and header fields.

As soon the `aname` Account is created in the CommuniGate Pro `domain.dom` Domain, mail starts to go to that Account automatically. You can copy all messages from the `aname` account on the old server to the `aname` Account on the new server and phase out the `aname` account on the old server.

## How can I relay E-mail and Signals for certain domains?

To allow your Server to relay all E-Mail and Signals to the friend.com domain place the following record into the [Router](#) table:

```
Relay: friend.com = friend.com@friend.com._via
```

Read the [Protection](#) section to learn the meaning of the Relay: prefix (you can omit it, or you may want to use the RelayAll: prefix instead).

If you want to relay E-mail and Signals for the friend.com domain, but they should go via some different `firewall.friend.com` server, use the following Router record:

```
Relay: friend.com = friend.com@firewall.friend.com._via
```

If you want to bypass the MX or SRV records and relay all E-mail and Signals to a certain IP address (specified explicitly or using a DNS A-record), then see the [Bypassing MX/SRV](#) section.

If you want your Server to act as a backup E-mail relay for certain domains, you can enable the `Relay to All Hosts We Backup` option in the [SMTP module](#) settings.
This is not a perfect solution, since anybody who can modify DNS records for certain domains can use your server as a backup relay for those domains.

## How can I send E-mail and Signals to a remote host bypassing its DNS MX/SRV records?

If your Server should send E-mail and Signals to the `target.domain` domain via the `relay.domain` relay server or proxy, you can specify the IP address of that relay with a [Router](#) record:
```
target.domain = target.domain@[11.22.33.44]
```

You may want to relay E-mail and Signals using DNS A-records instead of explicitly specified IP addresses:

```
Mail:target.domain = target.domain@relay.domain.25._via
Signal:target.domain = target.domain@relay.domain.5060._via
```

The [SMTP module](#) does not look at the MX records if the port number of a remote host is explicitly specified. By specifying the standard (25) SMTP port number, you tell the SMTP module to look for the relay.domain DNS A-record, and ignore its MX records.

The [SIP module](#) does not look at the SRV records if the port number of a remote host is explicitly specified. By specifying the standard (5060) SIP port number, you tell the SIP module to look for the relay.domain DNS A-record, and ignore its SRV records.

**Note:** You may want to add a `Relay:`, `NoRelay:` or `RelayAll:` prefix to these Router records.

---

# SMTP, SIP

## How can I forward mail to the other SMTP MTA on the same server?

You may want to have two different SMTP Servers (MTA) running on the same computer, but listening on either different port numbers or on different IP addresses.

To relay mail to the "sibling" server running on the port 26, you can redirect to the domain `other-port` if you put the following record into your [Router](#) table:

```
other-port = 127.0.0.1.26._via
```

To relay mail to the "sibling" server running on the port 25, but on a different IP address 11.22.33.44, you can redirect to the domain `other-ip` if you put the following record into your [Router](#) table:

```
other-ip = 11.22.33.44.25._via
```

For example, if all mail to the domain client57.com should go to the sibling server running on a different port, place the following records into the Router:

```
other-port = 127.0.0.1.26._via
Relay: client57.com = client57.com@other-port
```

or simply:

```
Relay: client57.com = client57.com@127.0.0.1.26._via
```

## How can my customer servers receive mail if they have dial-up connections? (ETRN)

Small sites may have dial-up connections only and they can be off-line most of the time. To provide better mail delivery to those sites, you should use your CommuniGate Pro server as their back-up mail relay. You should:

- create 2 MX records (in your DNS server) for the customer `domain.dom` domain name: a high priority record pointing directly to the customer server and a lower priority record pointing to your CommuniGate Pro server;
- configure your server to let it [relay mail](#) to the `domain.dom` domain;
- optionally include the `domain.dom` name into the [Hold Mail list](#) of your CommuniGate Pro [SMTP module](#);
- configure the customer server to send the wakeup `ETRN` commands to your server.

## How can I hold all client mail till their servers send ETRN?

If your client has a symmetric dial-on-demand link (i.e. a link that is brought up by the provider when there is any traffic to the client hosts), that client may want:

- to get all mail via your server instead of receiving mail directly, when each incoming message brings the connection link up;
- to receive mail from your server only when the client software issues the ETRN command, so your server will not bring the link up and try to relay the client mail as soon as it is received.

To serve such a customer (the `client.com` mail domain), you should:

- create a DNS A-record for the `mail.client.com` name, pointing to the IP address of the client server;

- create a DNS MX record for the `client.com` domain pointing to your CommuniGate Pro server; you should NOT include the `mail.client.com` name into the MX records for the `client.com` mail domain.
- create a record in the CommuniGate Pro Router:
  `client.com = mail.client.com._via`
- include the `mail.client.com` name into the [SMTP module](#) Hold Mail for Domains setting.

# How can my customer servers receive mail if they have dynamic IP addresses? (ATRN/PROP)

If a customer has a mail server and a dial-up connection with a dynamic IP address, the customer server cannot be listed in the DNS, because DNS records link domain names and fixed (static) IP addresses.

To deliver mail to those sites, you should configure your CommuniGate Pro server as their mail relay. Depending on the customer server capabilities, your can use either the `ATRN` or the Unified Domain-Wide Account (RPOP) method.

If the customer server supports the On-Demand Mail Relaying (ATRN) method, you should:

- create an MX record (in the your DNS server) for the customer `domain.dom` domain name; this record should point to your CommuniGate Pro server address;
- include the `domain.dom` name into the Hold Mail list of your CommuniGate Pro [SMTP module](#);
- create the `domain.dom` account in your CommuniGate Pro server Main Domain and assign some password to that account;
- configure the customer server to send the ATRN command to your server using `domain.dom` as the login (AUTH) name and the domain.dom account password as the AUTH password. If the customer software cannot send the ATRN command, it may send the TURN command, but only after it sends the AUTH command with the proper name and password.

If the customer server supports the Unified Domain-Wide Account method, you should:

- create an MX record (in the your DNS server) for the customer `domain.dom` domain name; this record should point to your CommuniGate Pro server address;
- create the `dd-customer` account (actual name is not relevant) in your CommuniGate Pro server Main Domain and assign some password to that account;
- add the following record to the CommuniGate Pro [Router](#):
  `domain.dom = dd-customer.local`
- configure the customer server to poll the `dd-customer` account on your CommuniGate Pro server;
- configure the customer server to use the `X-Real-To` header field (or other field you have specified in the [Local Delivery](#) module settings) as the "special header" containing the mail envelope information.

# How can my customers release mail to all their domains with one ETRN or ATRN?

Remote servers that use your CommuniGate Pro server as a back-up mail relay can serve multiple domains. Those servers usually send ETRN or ATRN commands specifying only one domain as the command parameter.

To let mail to all customer domains being released with one ETRN or ATRN command, you should enqueue mail sent to the customer "secondary" domains into the customer "main domain" queue.

If the remote server should receive mail for the domain1.dom, domain2.dom, and domain3.dom domains, but it sends ETRN or ATRN commands only for the domain1.dom domain, use the following Router domain-level records:

```
domain2.dom = domain2.dom@domain1.dom._via
domain3.dom = domain3.dom@domain1.dom._via
```

Mail to all customer domains will be placed into the domain1.dom queue, and if you want to hold that queue till the

ATRN/ETRN command is sent, include the domain1.dom name into the Hold Mail for Domains setting of the SMTP module.

---

# Rules

## How can I store all outgoing mail sent by all my users?

You may need to store all outgoing mail into a Mailbox in a system administrator or a security officer Account.

To copy mail sent from certain Domains, use a Server-wide [Rule](#):

| Data | Operation | Parameter |
|------|-----------|-----------|
| Return-Path | is | |
| **Action** | **Parameter** | |
| Store in | | |

The account `security` should already exist in the main domain, and the Mailbox `outgoing` should already exist in that account.

## How can I restrict to whom my users can send mail?

You may meed to let certain groups of users send mail only to other members of that group and/or to only certain addresses outside that group.

The simplest way to implement restrictions is to organize these groups of users into CommuniGate Pro Domains. If all users in the Domain dept1.company.dom (except the user boss) are allowed to send mail only to the users in the same Domain and to the `supervisor@hq.company.dom` address, then the following Server-wide Rule should be used:

| Data | Operation | Parameter |
|------|-----------|-----------|
| Return-Path | is | |
| Return-Path | is not | |
| Any Recipient | not in | |
| **Action** | **Parameter** | |
| Reject with | | |

---

# Mailboxes

## How can I create and use Shared Mailboxes?

A shared Mailbox is a Mailbox in Account x that can be used by a user (Account) y. Shared Mailboxes can be used for incoming mail processed by a group of people (sales department, support department, etc.). Shared Mailboxes can be used as an extremely fast and effective alternative to mail and distribution lists: the `announce` Mailbox in the

`marketing` account can be used to store all company announcements. If all employees have `read` access to that Mailbox, a single copy of each announcement becomes available to everybody.

To use a Shared Mailbox, two steps must be taken: first, potential users of the shared Mailbox should be granted access rights for that Mailbox. On the second step the user mailers should be configured to access shared Mailbox(es). Since these shared Mailboxes belong to a different account, they are called *foreign* Mailboxes.

First, the owner of the shared Mailbox should create a regular Mailbox within his/her account. It is useful to create a special account `public` and create shared Mailboxes in that account. To grant others access rights to the shared Mailbox, the account owner should use either a decent IMAP client that can deal with ACL (Access Control Lists) or the WebUser Interface. The WebUser Interface section describes how you can set the desired Mailbox Access Rights.

If a shared Mailbox is created inside the `public` account, it is useful to grant all Mailbox Access Rights to the real shared Mailbox owner, so the owner can perform all operations with that Mailbox without logging in as the user `public`.

To access shared Mailboxes, user mailers should be configured to display both the user account's own Mailboxes, and the available shared (foreign) Mailboxes. The most universal method is to use the account Mailbox Subscription list. This list is a simple set of Mailbox names, and both account's own Mailbox and foreign Mailbox names can be included into that list.

Many IMAP clients can only use the Mailbox Subscription list, but they cannot modify that list, or they do not allow a user to enter a foreign Mailbox name into that list. In this case IMAP users should use the WebUser Interface to fill their subscription lists. If a shared Mailbox `announce` has been created in the account `marketing`, users should put the `~marketing/announce` foreign Mailbox name into their subscription lists.

The domain administrator can use the Account Template to specify the initial Mailbox Subscription list, so all new accounts automatically get subscriptions to some shared Mailboxes.

When shared Mailboxes are included into the Account Subscription List, the users should configure their mail clients to display all Mailboxes listed in the Subscription List:

- WebUser Interface users should check that the Show All Subscribed Mailboxes Setting is selected.
- Microsoft® Outlook Express users should open the IMAP account Properties panel and enable the Advanced setting called `Only Show Subscribed Folders`. Since in this mode the Outlook Express mailer shows ONLY the Mailboxes listed in the account Mailbox Subscription list, the users should include their own Mailboxes (Sent, Drafts, etc.) into their Subscription lists.
- Netscape® Messenger users should open the IMAP Mail Server Properties panel and enable the Advanced setting `Show only subscribed folders`. Since in this mode the Messenger mailer shows ONLY the Mailboxes listed in the account Mailbox Subscription list, the users should include their own Mailboxes (Sent, Drafts, etc.) into their Subscription lists.
  The Messenger automatically scans the `public` account and displays its shared Mailboxes made available for the Messenger user. As a result, if all shared Mailboxes are created in the `public` account, Netscape Messenger users should not do anything with the Mailbox Subscription Lists.

Some clients (including Microsoft Outlook and Outlook Express) cannot display foreign Mailboxes even if those Mailbox names are included into the account subscription list. Users of these mailers can access foreign Mailbox via Mailbox aliases. They should use the WebUser Interface to specify aliases for foreign Mailboxes they want to access. If a shared Mailbox `announce` has been created in the account `marketing`, users should create the `mkt-announce` Mailbox alias for the `~marketing/announce` foreign Mailbox. Their IMAP clients will display the `mkt-announce` name and will provide access to the `~marketing/announce` Mailbox messages.

The domain administrator can use the Account Template to specify the initial Mailbox Aliases, so all new accounts automatically get a predefined set of Mailbox aliases for the specified shared Mailboxes.

# How can an Administrator clean User Mailboxes?

Sometimes a Server or Domain Administrator should be able to check user Mailboxes to clean or file user messages. This can be done without actually logging to the Server under that user name.

The Server Administrator with the [All Domains](#) access right has unlimited access rights to all Mailboxes in all Accounts.
The Domain Administrator with the `CanAccessMailboxes` access right has unlimited access rights to all Mailboxes in that Domain Accounts.

Administrators can use any decent IMAP, MAPI, or XIMSS client to access user Mailboxes. That client should be able to let a user enter a Mailbox name directly.
To open the INBOX Mailbox in the `username` Account, administrators should log in under their own names and tell the client to open the `~username/INBOX` Mailbox.

The [WebUser](#) Interface can be used for the same purpose. Administrators can log in under their own names, open the Subscription page and type the user Mailbox name in the Open Mailbox panel.

---

# Account File Storage

## How can I provide `username.domain.dom` Personal Web sites?

The standard URL for the username@domain.dom Account File Storage is `http://domain.dom/~username`.

You may want to provide better looking `http://username.domain.dom/` URLs for your Account personal Web sites. This feature is based on the method the Server uses to [route requests](#) sent to the HTTP User ports.

For users in the domain.dom secondary Domain, add the following records to the [Router](#):

```
*.domain.dom = *@domain.dom
<LoginPage%*@domain.dom> = *@domain.dom
```

If the domain.dom is your Main Domain, then add the following records:

```
*.domain.dom = *@fict
<LoginPage%*@fict> = *
```

These records route the `LoginPage@username.domain.dom` addresses to `username@domain.dom` addresses (or `username` addresses if domain.dom is the Main Domain).

Finally, you should update your DNS server to ensure that all `username.domain.dom` names point to your Server IP address.
You may want to use wildcard records (`*.domain.dom CNAME domain.dom`) if your DNS server supports them.

---

CommuniGate Pro

| Introduction | Features | History | How To | Help Me |

# Help Me

- **Security**
  - **Is my Server an open relay?**
  - **An Account was compromised and my server is being used for mass mailing. What can I do?**
- **WebAdmin**
  - **I have rerouted the Postmaster account and now I cannot log in as the Postmaster.**
  - **I have deleted the Postmaster account.**
  - **I have created a secondary Domain and now I cannot log into WebAdmin.**
  - **When I try to log in, I get the "access from your network is denied" error.**
- **SMTP Receiving**
  - **My Server does not accept mail from my Web script/applet**.
- **SMTP Sending**
  - **My Server cannot send mail to some host using SSL/TLS**.
- **Access**
  - **WebUser connections return the pink page saying "we do not provide Web Access to this domain".**
  - **WebUser sessions are disconnected almost immediately after login.**
  - **What does the "unassigned local network address" error mean?**
- **Directory**
  - **Microsoft LDAP (Outlook and Outlook Express) users cannot find Directory records.**
  - **Attempts to update Account Settings result in the `directory record with the specified DN is not found` error.**
- **Date and Time**
  - **The time stamps in messages sent or received with CommuniGate Pro are several hours off.**
- **Logs**
  - **Every time I access the WebAdmin interface, a Failure-type ROUTER record appears in the Log**.
  - **What do these `DNR-16538(xxx.xx.x.xx.rss.mail-abuse.org) A:host name is unknown` records mean?**
- **Miscellaneous**
  - **What is that non-standard UDP port the CommuniGate Pro Server opens on my system?**.
  - **How can I make my `formmail`-type CGI work with CommuniGate Pro?**.

This section lists the most common problems with the CommuniGate Pro installations, and it provides the suggestions that should help you to solve those problems.

# Security

## Is my Server an open relay?

Open Relay is an SMTP (or SIP) server configured in such a way that it allows anyone on the Internet to send e-mail (or make calls) through it, not just mail destined to or originating from known users. If you receive a lot of mail/spam from unknown origin, but the targets are your local users, then it has nothing to do with relaying; relaying means sending through your server to external targets.

With the default settings CommuniGate Pro is configured NOT to be an open relay, it relays only e-mails (calls) submitted by senders who had authenticated.
Relaying for non-authenticated senders is possible if the sender had connected from an address from the Client IP Addresses list, so make sure there are no excessive addresses there; ideally that list should be empty and all your users must authenticate when sending. If you receive all mail from a gateway - do NOT add the gateway address to the `Client IP Addresses` list, but add it to UnBlacklistable (White Hole) IP Addresses list.

In SMTP Relaying page make sure:

- the "Relay to any IP Address: If Received from:" is set to `clients` or `nobody`
- the "Relay to Client IP Addresses: If Sent to:" is set to `simple` or `none`
- the "Relay to Hosts We Backup:" is `disabled`

## An Account was compromised and my server is being used for mass mailing. What can I do?

Someone had learned/guessed the password of an Account and uses that Account to send spam. Note that this case has nothing to do with open relaying.

Open the Mail page in the WebAdmin Monitors realm, and open the Queue page. There you should see a lot of messages with similar size and contents.

- Open one of such messages to learn the compromised Account name and the sender's IP address.
- Click the Reject All Sender's Messages button.
- Open the compromised Account Settings page.
  - Reset or disable the password to prevent new logins.
  - Temporary disable the `Mail` service to stop submitting mail from existing logins.
- Use the Reject All Sender's Messages button to clean the Queue. The messages which are being processed may not be rejected, so you may need to repeat this step several times.

In order to lower the chances of the users' passwords becoming compromised:

- Make sure all users use encrypted connections (SSL/TLS) when commuicating with the server. That will prevent hackers from learning passwords via network sniffers.
- Force users to have enough long and complex passwords which cannot be guessed easily.
- Force Two-factor Authentication, if possible.
- Impose tighter limits in
  - "Failed Logins Limit" in the Account Settings page
  - Temporarily Blocked IP Addresses in Settings->Network->"Blacklisted IPs" page
  to prevent brute-force attacks.
- Enable the Hide 'Account Unknown' messages to hinder address harvesting.

To reduce the damage caused by compromised Accounts, and to make them to be less attractive for hackers:

- Impose tighter limits for "Outgoing Mail Limit", "Outgoing Recipients Limit" and "Max Recipients per Message" in Outgoing Mail Transfer Settings.
  That will reduce the rate a hacker will be able to send messages.
- Impose "'From' Address Restrictions" and "'From' Name Restrictions" in Outgoing Mail Transfer Settings.
  That will give the hackers less freedom for spoofing the message origin.
- If your customers are to use WebMail/Pronto only and no external SMTP clients, then in the Enabled Services disable the `Relay` service. That will disallow hackers to use SMTP which is the most convenient way to submit messages.

---

# WebAdmin

## I have rerouted the Postmaster account and now I cannot log in as the Postmaster

CommuniGate Pro applies routing rules not only to addresses in incoming messages, but to all addresses it processes. If you have rerouted the `postmaster` account to some other account `abc`, then all attempts to log in as the `postmaster` will cause the Server to try to open the `abc` account. If you provide the correct password (i.e. the `abc` account password), you will be able to log in, but you will have the access rights granted to the `abc` account, not to the `postmaster` account.

You still can log into the `postmaster` account even if the `postmaster` name is redirected to a completely different address. Use the following name instead of the `postmaster` name:

abcd@postmaster.local
This address is always routed to the account `postmaster`. Use the regular `postmaster` account password with this string.

For more details on the `.local` routing, check the Local Delivery Module section.

## I have deleted the Postmaster account

If you have deleted the `postmaster` account, stop the Server and start it again.

If the CommuniGate Pro Server does not find the `postmaster` account during the startup process, it creates a new one. Check the `postmaster` account files to get the new `postmaster` password, in the same way you used when you installed the CommuniGate Pro Server.

## I have created a secondary Domain and now I cannot log into WebAdmin

When you connect to CommuniGate Pro via a browser, the Server checks the domain name you have specified in the browser URL. If that name matches the name of one of your Secondary Domains, the WebAdmin Interface of that Domain is opened, rather than the Server WebAdmin Interface.

To open the Server WebAdmin Interface, use the Main Domain Name in your browser URL. If that name does not have a DNS A-record or its record points to a different server, use the Server IP Address in the browser URL.

If all Server IP Addresses were assigned to secondary Domains, you can try to use ANY domain name that points to the CommuniGate Pro Server, and does not match any of the Secondary Domain names.

If all Server IP Addresses were assigned to secondary Domains and all DNS domain names pointing to your server are

names of your secondary Domains or secondary Domain Aliases, then use the following URL:

```
http://sub.domain.com:8010/MainAdmin
https://sub.domain.com:9010/MainAdmin
```

where *sub.domain.com* is any name pointing to your server computer or any of its IP addresses.

## When I try to log in, I get the "access from your network is denied" error

Open the Network pages in the WebAdmin Settings realm, and open the Client IPs page. The Logins from Non-Client IP Addresses option is set to `prohibit`, so users can connect to the Server only from the addresses listed in the Client IP Addresses field (on the same page).

If the Client IP Addresses field was left empty, you still can connect to the Server if you launch your browser on the Server computer itself, and connect locally, using the `http://127.0.0.1:8010` URL.

If you have not entered anything into the Client IP Addresses field, or if you cannot connect from the IP Addresses listed in that field, and you cannot connect to the server locally, using the `http://127.0.0.1:8010` URL, then:

- stop the CommuniGate Pro Server;
- open the `{base}`/`Settings/IPAddresses.settings` file and change the ClientOnly option from `YES` to `NO`, and save the updated file.
- start the CommuniGate Pro Server again.

# SMTP Receiving

## My Server does not accept mail from my Web script/applet

When the SMTP module receives messages, it tries to route the address specified in the Mail From command (the message 'Return-Path' address). If the domain name in that address is a name of the Server local Domain and the specified Account (or other Object) is not found in that Domain, the Router returns an error code and the SMTP module refuses to accept the message.

You should reconfigure your script/applet to use either an empty Return-Path (`<>`) for generated messages, or to use an E-mail address of some existing Account. If the script/applet cannot be reconfigured, you can create an Alias for any existing Account.

If, for example, your script/applet submits messages to your server with the <webform@mydomain.com> Return-Path address, and you do not have the webform Account in the mydomain.com Domain, you may want to create the webform alias for the postmaster Account. If delivery of a submitted message fails, the error report will be sent to the postmaster Account.

# SMTP Sending

## My Server cannot send mail to some host using SSL/TLS

When the CommuniGate Pro SMTP module connects to a mail host/relay and tries to establish a secure (SSL/TLS) connection, it receives the host Certificate and check the name in that certificate. That name should match either the name of the domain the mail should go to, or the MX relay name for that domain name.

When a remote server hosts several domains on the same IP address, it always sends out only one certificate, because the server cannot learn to which domain the incoming messages will go to and thus it cannot present the Certificate for

that particular domain. As a result, your (sending) server may refuse to proceed.

If the server mainhost.com also hosts client1.com and client2.com domains, and the MX records for all 3 domains point to the same name and to the same IP address on that server, the server will always present only one Certificate - usually, the mainhost.com Certificate.

To allow your CommuniGate Pro Server to send mail securely to client1.com and client2.com domains, you should specify 2 Domain-level [Router](#) records:

```
client1.com = client1.com@mainhost.com._via
client2.com = client1.com@mainhost.com._via
```

These records will place mail to client1.com and client2.com domains into the mailhost.com SMTP queue. You should place the mainhost.com name into the Send Encrypted list of the SMTP module, and the server will connect to the mailhost.com server, check its certificate (it should contain either the mailhost.com name or the name of the relay the SMTP module connected to), and then the SMTP module will establish a secure (SSL/TLS) connection with that server and it will send mail to recipients in the client1.com and client2.com domains via that secure connection.

---

# Access

## WebUser connections return the pink page saying "we do not provide Web Access to this Domain"

It is very important to understand that the domain name `something.com` and `mail.something.com` are completely different domain names. If your CommuniGate Pro Server has the main Domain `mycompany.dom`, and you are trying to connect to it by typing `http://mail.mycompany.com:8100` in your Web browser, you will get the page saying that the CommuniGate Pro Server does not provide access to the `mail.mycompany.com` Domain.

In most cases, you want the domain names `mail.mycompany.com`, `webmail.mycompany.com`, etc. to be just other names (aliases) of the `mycompany.com` CommuniGate Pro Domain. To specify this, open the `mycompany.com` Domain Settings page and find the Aliases table. In an empty field, enter the `mail.mycompany.com` name and click the Update button. Now the CommuniGate Pro Server will know that `mail.mycompany.com` domain name is just a different name for the `mycompany.com` Domain it serves. Connection requests specifying the `mail.mycompany.com` domain name will connect to the `mycompany.com` CommuniGate Pro Domain, and messages sent to a `username@mail.mycompany.com` address will be delivered to the account *username* in the `mycompany.com` domain.

**Note:** The WebAdmin interface opens the Server Administrator Interface if the name specified in the browser URL is not a CommuniGate Pro Domain name. This is why connections to the WebAdmin port (8010) can work, while the connections to the WebUser port (8100) return the "pink page".

## WebUser sessions are disconnected almost immediately after login

When a user connects to your server via a "multi-homed HTTP proxy" (used by large ISPs such as AOL), TCP connections come to the CommuniGate Pro Server from several different IP addresses of those proxy servers. If the `Require Fixed Network Address` option is enabled in the Account WebUser Preferences, user browser connections can be rejected. Disable the `Require Fixed Network Address` option for those users that connect via "multi-homed proxy" servers. If most of your users connect via those proxy servers, you may want to disable this setting in the Domain Account Defaults or in the All-Server Account Defaults.

## What does the "unassigned local network address" error mean

Your CommuniGate Pro server computer has one or several IP (network) addresses assigned to it. Those addresses

can be assigned to CommuniGate Pro Domains, and the Domains WebAdmin page shows all Domains with the IP addresses assigned to them.

Usually, the Main Domain has the Assigned IP Addresses setting set to All Available, so all IP Addresses not assigned to secondary Domains are automatically assigned to the Main Domain. If none of your Domains has the Assigned IP Addresses setting set to All Available, then some of your Server IP addresses may be not assigned to any Domain.

When a user connects to the server using a POP or IMAP client and provides just the account name (without the domain name), or when a secure (SSL/TLS) connection has to be established, the CommuniGate Pro Server takes the local IP address the user has connected to and tries to find the Domain that address is assigned to. If that IP address is not assigned to any CommuniGate Pro Domain, then the "unassigned local network address" error is generated.

Open the WebAdmin Settings->General page to see all the Local IP Addresses of your Server. You may have to click the Refresh button to see all addresses. The unassigned IP Addresses are displayed in red.

# Directory

## Microsoft LDAP (Outlook and Outlook Express) users cannot find Directory records

Most of LDAP clients (including the Microsoft Outlook products) contain a setting specifying the Directory subtree that should be used for search operations. In Outlook Express, this setting can found in Tools->Accounts->Directory Service, on the Advanced stub. It is called Search Base and it should contain the DN for the user domain (by default, that DN is `cn=domainname`).

If this setting field is left empty, Outlook products silently replace it with the `c=country_code` string, and search operations fail (unless your Directory has the `c=country_code` subtree).

If you do want to search the entire Directory with an Outlook product, enter the word `top` into the Search Base setting field.

## Attempts to update Account Settings result in the `directory record with the specified DN is not found` error

This error appears when the [Directory Integration](#) option is enabled. This option tells the CommuniGate Pro Server to update the Account record in the Central [Directory](#) every time the Account Settings are updated. If the Directory does not contain a record for that account, the error message is returned. Account records may be missing in the Directory if the Accounts were created when the Directory Integration option was disabled.

To fix the problem, open the Domain Settings and find the [Directory Integration](#) panel. Click the Delete All button. It will remove all Domain object records from the Directory. Then click the Insert All button. The CommuniGate Pro Server will create a Directory record for the Domain, and then it will create Directory records for all Domain Objects (Accounts, Groups, Mailing Lists).

**Note:** if the Domain contains more than 100,000 Accounts, the Insert All operation can take several minutes.

# Date and Time

## Time stamps in messages sent or received with CommuniGate Pro are

### several hours off

This problem is caused by an incorrect Time Zone setting on the server and/or on the client machines. To check the Time Zone setting value on the server machine, open the General page in the Settings realm of the CommuniGate Pro WebAdmin Interface. The Server Time field should contain the correct Date and Time values **and** the correct Time Zone value: -0800 means '8 hours behind the GMT', +0800 means '8 hours ahead of GMT'.

If the Time Zone value is incorrect, fix the OS settings that specifies that value, and re-open the General page to verify the Time Zone value.

# Logs

## Every time I access the WebAdmin interface, a Failure-type ROUTER record appears in the Log

The WebAdmin interface adds the `LoginPage@` string to the domain name you specify in your browser URL field and tries to route the resulting address as any other E-mail address. If routing fails, the WebAdmin Interface defaults to the main domain and to the Server WebAdmin Interface, but the failure record appears in the Router Log:

```
ROUTER failed to route 'LoginPage@mail'
```
Usually this happens when you use a non-qualified domain name (like `mail`) instead of the qualified domain name (`mail.mycompany.com`). You should either use the qualified domain name in your browser URLs, or you should add the `mail` Domain Alias to the `mail.mycompany.com` CommuniGate Pro Domain.

## What do these `DNR-16538(xxx.xx.x.xx.rss.mail-abuse.org) A:host name is unknown` records mean?

When your SMTP module uses RBLs to check the IP address of the server that tries to send any mail to your server, it converts that server aa.bb.cc.dd IP Address into the dd.cc.bb.aa.*rbl-server-name* domain name, and tries to resolve this name using the DNS system. If the sending server is not a known offender, and its address is not included into the RBL database, this composed domain name will NOT exist in the DNS system, and the DNR module will report this with a Problem-level Log record.

If you use RBL servers, you may want to restrict the DNR module Log Level to Major & Failures events only.

# Miscellaneous

## What is that non-standard UDP port the CommuniGate Pro Server opens on my system?

This is a DNR (Domain Name Resolver) socket. The port number is selected by the OS, and it can change if you restart the CommuniGate Pro Server. This socket is used to send requests (UDP packets) to DNS servers and to receive responses from those servers.

Other applications (servers, browsers, etc.) use the same type of sockets to resolve domain names, but they usually open and close those UDP sockets quickly, so you may not notice them in your `netstat` output. CommuniGate Pro opens the DNR UDP socket when it starts, and uses that socket for all DNR requests, closing the socket only when the Server shuts down.

# How can I make my `formmail`-type CGI work with CommuniGate Pro?

`Formmail` and similar CGIs are used to send E-mail messages from regular Web Server HTML forms. Implemented in the form of a [Perl](#) script, these CGIs use the legacy `sendmail` program to send the composed messages.

On most platforms, CommuniGate Pro software installer does not replace the legacy `sendmail` program, though the package does contain the `sendmail` replacement program. In order to use that program, you should modify your Perl script: you should find all references to the `sendmail` program (usually the default path used is `/usr/sbin/sendmail`), and replace them with the `{application directory}/sendmail` references.

For example, if CommuniGate Pro and your CGI are installed on a MacOS X system, where the CommuniGate Pro *application directory* is `/usr/sbin/CommuniGatePro/`, the CGI script `/usr/sbin/sendmail` strings should be replaced with the `/usr/sbin/CommuniGatePro/sendmail` strings.

---

| **Installation** | **QuickStart** | **Migration** |
|---|---|---|

# Installation

- **Installation**
  - **Installing on a Sun Solaris System**
  - **Installing on a Linux System**
  - **Installing on a MS Windows System**
  - **Installing on a MacOS X (Darwin) System**
  - **Installing on a FreeBSD System**
  - **Installing on a NetBSD System**
  - **Installing on an OpenBSD System**
  - **Installing on a BSDI BSD/OS System**
  - **Installing on an AIX System.**
  - **Installing on an HP/UX System**
  - **Installing on a Tru64 System**
  - **Installing on an SGI IRIX System**
  - **Installing on an SCO UnixWare System**
  - **Installing on an SCO OpenServer System**
  - **Installing on an IBM OS/400 System**
  - **Installing on an OpenVMS System**
  - **Installing on a QNX System**
  - **Installing on an IBM OS/2 System**
  - **Installing on a MacOS Rhapsody System**
  - **Installing on a BeOS System**
- **Initial Configuration**
- **Upgrading to a Newer Version**
- **Moving to a New Hardware Server**

You should download the CommuniGate Pro software either from the CommuniGate Systems Web/FTP site, or from any authorized mirror site. Make sure you have the latest version of the software, and that the downloaded version is designed to work on the selected platform.

Install the Server following the instructions below, and then proceed with Initial Configuration.

---

# Installation

On all systems, the CommuniGate Pro uses 2 directories (folders):

- the *application directory* containing the Server application and supplementary files (as Web Interface page templates). Files in this folder are not modified when the system runs.
- the *base directory* containing Account data, Settings, Queued messages, Logs, and other Server data.

---

## Installing on a Sun Solaris System

- Make sure you are running Solaris version 8 or better.
- Log in as a super-user (root).
- Unpack the CommuniGate Pro archive with the `gtar` command (or with the `gunzip` and `tar` commands):
  ```
  gunzip CGatePro-Solaris-version.tar.gz
  tar -xpf CGatePro-Solaris-version.tar
  ```
- Install the CommuniGate Pro package:
  ```
  pkgadd -d .
  ```
  The CommuniGate Pro software will be installed in the `/opt` directory.
- If your system was running `sendmail`, `Postfix` or any other SMTP server, stop that server and remove that server start-up script from the `/etc/rcn/` directory, so the system will not start that other SMTP server automatically.
- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- The Installer creates a symbolic link `/bin/cgmail` for the command line mode mail program to use with the CommuniGate Pro system.
- The Installer creates a startup script `/etc/init.d/STLKCGPro.init`, and the symbolic link `/etc/rc2.d/S88CommuniGate` for that script.
- The Installer creates the "base directory" `/var/CommuniGate` and the Server uses it by default. You can move the "base directory" to any other location. In this case, open the `/etc/init.d/STLKCGPro.init` file and update it.
- Restart the system or launch the start-up script manually:
  ```
  /etc/init.d/STLKCGPro.init start
  ```
- Proceed with [Initial Configuration](#).

---

# Installing on a Linux System

- Log in as a super-user (root).
  - Using the Red Hat Package Manager (the .rpm file):
    ```
    rpm -i CGatePro-Linux-version.rpm
    ```
  - Using the Debian Package Manager (the .deb file):
    ```
    dpkg -i CGatePro-Linux-version.deb
    ```
  - Using other systems (the .tgz file):
    ```
    tar -xzf CGatePro-Linux-version.tgz
    cd CGateProSoftware
    sh install.sh
    ```
  The CommuniGate Pro software will be installed in the `/opt` directory.
- The installer will create the `/etc/rc.d/init.d/CommuniGate` startup script. To make the CommuniGate Pro Server start and stop automatically when the system starts up and shuts down, the installer adds the startup file links to the `/etc/rc.d/rcn.d` directories.
- If your system was running some standalone SMTP server/MTA (such as Postfix, Exim or sendmail), you need to stop and disable that server.
  For example, you can use this command to disable sendmail (need to restart the server computer to take effect):
  ```
  /sbin/chkconfig sendmail off
  ```
  or these commands to stop and remove Postfix:
  ```
  systemctl stop postfix
  yum remove postfix
  ```
- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file (or put the hash `#`) symbol into the first position of those lines).
- The Installer renames the `/bin/mail` program into the `/bin/LegacyMail`. If you decide to uninstall the CommuniGate Pro system, the legacy mail program will be renamed back to `/bin/mail`.
- The Installer creates the new `/bin/mail` application - a drop-in substitution for the legacy `mail` program.
- The Installer creates the "base directory" `/var/CommuniGate` and the Server uses it by default. You can move the "base directory" to any other location. In this case, open the `/etc/rc.d/init.d/CommuniGate` file and update it.
- Restart the system or launch the start-up script manually:
  ```
  /etc/rc.d/init.d/CommuniGate start
  ```
- Proceed with [Initial Configuration](#).

**Note:** some older versions of Linux (such as RedHat 9.0, SuSE 9.1 and some other distributions) used a very unstable version on the NPTL p-threads library.
To provide a workaround for these versions of the Linux OS, the CommuniGate Pro startup script uses the `LD_ASSUME_KERNEL=2.4.1` command to make the Linux linker use the old, more stable version of the p-threads library.

**Note:** When the old p-threads library is used, each CommuniGate Pro thread is seen as a separate process when you use the `ps` and `top` system utilities. This is normal: all those "processes" are actually CommuniGate Pro Server threads, and they share all their resources - VRAM, file descriptors, etc.

**Note:** Linux kernels prior to 2.6.13 have critical flaws in their NFS client implementations. If you plan to use the Linux OS as your Dynamic Cluster backends, make sure that your kernel version is 2.6.13 or higher.

## Installing on a MS Windows System

- Use any "unzip"-type tool to unpack the `CGatePro-Win32-Intel-version.zip` file. **The tool should be able to use long file names.**
- Some "unzip" applications have the Install option, use that option if available. If the Install option is not available, unpack the archive. The unpacked directory should contain the CommuniGate Pro application folder and the Installer.exe application. Launch the Installer.exe application.
- If the CommuniGate Pro software is running, the Installer asks your permission to stop it.
- The Installer asks you where to place the CommuniGate Pro application folder, and where to create the "base" directory. If a previous version of CommuniGate Pro has been already installed, the Installer shows the locations used, and the Install button is renamed into the Update button.
- Click the Install/Update button to copy the CommuniGate Pro software into the selected location. If the CommuniGate Pro "base" folder does not exist, the Installer creates an empty folder in the selected location.
- The information about the selected locations is stored in the System Registry.

    Windows NT/200x/XP/Vista/Windows 7 and newer
    The CommuniGate Pro Messaging Server (the `CGStarter.exe` application) is registered as a *service* that starts automatically when the system starts. This small application starts the `CGServer.exe` - the CommuniGate Pro Server application. The Installer asks if you want to start the Server now.

    **Note:** you should use the Services control panel to verify or change the `Log On as` name for the CommuniGate Pro service. That name should have the `Act as part of the operating system` Windows NT privilege. If the CommuniGate Pro Server does not have this privilege, not only it will fail to authenticate users using the Windows NT password system, but an attempt to use an incorrect password may cause the server to crash. This problem is fixed in the Windows NT Service Pack 4.

    **Note:** if your server should serve 100 accounts or more, check the description of the TIME_WAIT problem and follow the instructions to decrease the NT TIME_WAIT time interval.

    Windows 95/98/ME
    The CommuniGate Pro (the `CGServer.exe` application) is added to the RunServices Registry key, so the server is started up when the system starts. Restart the Windows 9x system to start the server.

    **Note:** Unlike Windows 98/ME, the Windows 95 system does not have the "WinSock2" library installed. Download this library (.dll) from the http://www.microsoft.com and install it before you try to launch the CommuniGate Pro Server.

- Launch the Server and proceed with Initial Configuration.

You can also start the CommuniGate Pro server manually, as a "console application", by launching the CGServer.exe file. If started without parameters, the Server creates the `C:\CommuniGatePro` folder and uses it as its "base" folder". If you want to use any other location, use the --Base command line parameter:

`CGServer.exe --Base D:\OtherDirectory`

# Installing on a MacOS X (Darwin) System

- Make sure you are running MacOS X version 10.5 or better.
- Log in as a user with the administrative rights.
- Unpack the CommuniGate Pro archive using any uncompressing utility, or start the Terminal application and use the shell `tar` command:

   `tar xzpf CGatePro-Darwin-platform-version.tgz`

   the CommuniGate.pkg *package directory* will be created in the current directory.
- Install the software by double-clicking the CommuniGate.pkg icon.
  The CommuniGate Pro software will be installed in the `/usr/local/` directory.
- Note: The Installer creates the startup directory `/Library/StartupItems/CommuniGatePro`, so the CommuniGate Pro Server will auto-start on the MacOS X System older than version 10.10. The Installer creates also `/Library/LaunchDaemons/CommuniGatePro.plist` item to control automatic start on newer versions of MacOS X.
- Note: The Installer packs the startup directory `/System/Library/StartupItems/Sendmail` into the `/System/Library/StartupItems/Sendmail.tar`, and the `/System/Library/StartupItems/Postfix` directory into the `/System/Library/StartupItems/Postfix.tar` archive, so the legacy `sendmail` and `postfix` daemons will not auto-start. If you decide to uninstall CommuniGate Pro, these archives will be unpacked and the legacy startup directories will be restored.
- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- The Installer renames the `/usr/bin/mail` program into the `/usr/bin/LegacyMail`. If you decide to uninstall the CommuniGate Pro system, the legacy mail program will be renamed back to `/usr/bin/mail`.
- The Installer creates the new `/usr/bin/mail` application - a drop-in substitution for the legacy `mail` program.
- The Installer creates the "base directory" `/var/CommuniGate` and the Server uses it by default. You can move the "base directory" to any other location. In this case, open the `/Library/StartupItems/CommuniGatePro/CommuniGatePro` file and update it.
- Restart the MacOS X System.
- Proceed with [Initial Configuration](#).

# Installing on a FreeBSD System

There are two CommuniGate Pro packages: one for FreeBSD 9.x (supporting FreeBSD 9.x versions), one - supporting FreeBSD 10.x and later versions.

- Log in as a super-user (root).
- Install the CommuniGate Pro package. FreeBSD 9.x:

   `pkg_add CGatePro-FreeBSD9-version.tgz`

   FreeBSD 10.x:

   `pkg add CGatePro-FreeBSD10-version.txz`

   The CommuniGate Pro software will be installed in the `/usr/local/sbin` directory.
- If your system was running `Postfix`, `sendmail` or any other SMTP server, stop that server and modify the OS start-up scripts so the system will not start that other SMTP server automatically.
- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- The Installer creates a script `/usr/local/etc/rc.d/CommuniGate.sh`, so the CommuniGate Pro Server is started automatically when the FreeBSD system starts.
- The Installer creates a symbolic link `/bin/cgmail` for the command line mode mail program to use with the

CommuniGate Pro system.

- The Installer creates the "base directory" `/var/CommuniGate` and the Server uses it by default. You can move the "base directory" to any other location. In this case, open the start-up script file and update it.
- Restart the system or launch the start-up script manually:
  `/usr/local/etc/rc.d/CommuniGate.sh start`
- Proceed with [Initial Configuration](#).

---

# Installing on a NetBSD System

- Make sure you are running NetBSD version 2.0 or better.
- Log in as a super-user (root).
- Install the CommuniGate Pro package:
  `pkg_add CGatePro-NetBSD-version.tgz`
  The CommuniGate Pro software will be installed in the `/usr/pkg` directory.
- If your system was running `Postfix`, `sendmail` or any other SMTP server, stop that server and modify the OS start-up scripts so the system will not start that other SMTP server automatically.
- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- The Installer creates a startup script `/etc/rc.d/CommuniGate`, so the CommuniGate Pro Server is started automatically when the NetBSD system starts.
- The Installer creates a symbolic link `/bin/cgmail` for the command line mode mail program to use with the CommuniGate Pro system.
- The Installer creates the "base directory" `/var/CommuniGate` and the Server uses it by default. You can move the "base directory" to any other location. In this case, open the start-up script file and update it.
- Restart the system or launch the start-up script manually:
  `/etc/rc.d/CommuniGate start`
- Proceed with [Initial Configuration](#).

---

# Installing on an OpenBSD System

- Make sure you are running OpenBSD version 3.4 or better.
- Log in as a super-user (root).
- Install the CommuniGate Pro package:
  `pkg_add CGatePro-OpenBSD-version.tgz`
  The CommuniGate Pro software will be installed in the `/usr/local/sbin` directory.
- If your system was running `Postfix`, `sendmail` or any other SMTP server, stop that server and modify the OS start-up scripts so the system will not start that other SMTP server automatically.
- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- The Installer adds a reference to the CommuniGate Pro startup script to the `/etc/rc.local file`, so the CommuniGate Pro Server is started automatically when the OpenBSD system starts.
- The Installer creates the `mail` group if it was absent.
- The Installer creates a symbolic link `/bin/cgmail` for the command line mode mail program to use with the CommuniGate Pro system.
- The Installer creates the "base directory" `/var/CommuniGate` and the Server uses it by default. You can move the "base directory" to any other location. In this case, open the start-up script file (`/usr/local/sbin/CommuniGate/Startup`) and update it.
- Restart the system or launch the start-up script manually:
  `/usr/local/sbin/CommuniGate/Startup start`
- Proceed with [Initial Configuration](#).

---

# Installing on a BSDI BSD/OS System

- Make sure you are running BSD/OS version 4.0.1 or better.
- Log in as a super-user (root).
- Unpack the CommuniGate Pro archive with the `gunzip` and `tar` commands:
  ```
  gunzip CGatePro-BSDI-Intel-version.tar.gz
  tar -xpf CGatePro-BSDI-Intel-version.tar
  ```
- Install the CommuniGate Pro package:

  `installsw -c /.../CGatePro` (use the full path to the CGatePro directory)

  The CommuniGate Pro software will be installed in the `/usr/local/sbin` directory.
- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- If your system was running sendmail or other mail transfer agent, remove the lines describing those servers from the `/etc/rc` file.
- The Installer adds a reference to the CommuniGate Pro startup script to the `/etc/rc.local file`, so the CommuniGate Pro Server is started automatically when the BSDI BSD/OS system starts.
- The Installer creates a symbolic link `/usr/bin/cgmail` for the command line mode mail program to use with the CommuniGate Pro system.
- The Installer creates the "base directory" `/var/CommuniGate` and the Server uses it by default. You can move the "base directory" to any other location. In this case, open the start-up script file (`/usr/local/sbin/CommuniGate/Startup`) and update it.
- Restart the system or launch the start-up script manually:
  ```
  /usr/local/sbin/CommuniGate/Startup start
  ```
- Proceed with [Initial Configuration](#).

---

# Installing on an AIX System

- Make sure you are using the AIX System version 4.3 or better.
- Log in as a super-user (root).
- Unpack the CommuniGate Pro archive with the `compress` command:
  ```
  compress -d CGatePro-AIX-PPC-version.bff.Z
  ```
- Use either the `installp` command, or the `smit` utility, or the `smitty` utility to install the software.
- The installation script creates a startup script `/etc/rc.cgpro` and updates the `/etc/inittab` file to start the CommuniGate Pro server on run level 2.
- The startup script creates the "base directory" `/var/CommuniGate` and the Server uses this directory by default. You can move the "base directory" to any other location. In this case, open the `/etc/rc.cgpro` script file and update it.
- If your system was running `Postfix`, `sendmail` or any other SMTP server, stop that server and modify the OS start-up scripts so the system will not start that other SMTP server automatically.
- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- Restart the system or launch the start-up script manually:
  ```
  /etc/rc.cgpro start
  ```
- Proceed with [Initial Configuration](#).

---

# Installing on an HP/UX System

- Make sure you are running HP/UX version 11 or better.
- Apply all patches available for HP-UX 11.00 or at least all threads-related patches.
- Set the following kernel parameters:
  - set the `maxdsiz` kernel parameter to 100MB or more.
  - set the `max_thread_proc` (max number of threads per process) to 1024 or more.

set the `ncallout` parameter (max number of pending timeouts) to 1024 or more.
  ○ set the `maxfiles` parameter to 1024 or more.
- Log in as a super-user (root).
- Unpack the CommuniGate Pro archive with the `gunzip` and `tar` commands:
  ```
  gunzip -xzpf CGatePro-HPUX-platform-version.tar.gz
  tar -xf CGatePro-HPUX-platform-version.tar
  ```
  The `CGatePro.depot` directory should appear in the current directory.
- Install the CommuniGate Pro package:

  `swinstall -s `pwd`/CGatePro.depot` (you should use the absolute path for the unpacked `CGatePro.depot` directory)

  The CommuniGate Pro software will be installed in the `/opt/CommuniGate` directory.
- If your system was running `sendmail` or any other SMTP server, stop that server and modify the OS start-up scripts so the system will not start that other SMTP server automatically.
- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- The server startup script is created as `/sbin/init.d/CommuniGate` and its symbolic links `/sbin/rc2.d/S80CommuniGate` and `/sbin/rc1.d/K80CommuniGate` are automatically created.
- The Server uses `/var/CommuniGate` as its "base directory" by default. You can move the "base directory" to any other location. In this case, open the `/sbin/init.d/CommuniGate` script file and update its `BASEDIRECTORY` parameter.
- The symbolic link `/bin/cgmail` is created for the CommuniGate Pro "mail" program.
- Restart the system or launch the start-up script manually:
  ```
  /sbin/init.d/CommuniGate start
  ```
- Proceed with [Initial Configuration](#).

# Installing on a Tru64 (Digital Unix) System

- Make sure the Tru64 version 5.0 or better is installed.
- Log in as a super-user (root).
- Unpack the CommuniGate Pro archive with the `gtar` command (or with the `gunzip` and `tar` commands):
  ```
  gtar -xzf CGatePro-Tru647-platform-version.tar.gz.
  ```
- Install the CommuniGate Pro package:
  ```
  /usr/sbin/setld -l CGatePro.pkg
  ```
  The CommuniGate Pro software will be installed in the `/usr/opt/` directory.
- If your system was running `sendmail` or any other SMTP server, stop that server and modify the OS start-up scripts so the system will not start that other SMTP server automatically.
- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- The Installer creates the symbolic link `/sbin/init.d/CommuniGate` for the server startup script.
- The Installer also creates the `/sbin/rc0.d/K10CommuniGate`, `/sbin/rc2.d/K10CommuniGate`, and `/sbin/rc0.d/S80CommuniGate` startup symbolic links.
- The Server uses `/var/CommuniGate` as its "base directory" by default. You can move the "base directory" to any other location. In this case, open the `/usr/opt/CGPversion/startup` script file and update its `BASEFOLDER` parameter.
- Restart the system or launch the start-up script manually:
  ```
  /sbin/init.d/CommuniGate start
  ```
- Proceed with [Initial Configuration](#).

# Installing on an SGI IRIX System

- Make sure you are using the IRIX System version 6.5 or better.
- Log in as a super-user (root).
- Install the CommuniGate Pro package:

```
        inst -f CGatePro-IRIX-MIPS-version.tardist
```
or
```
        swmgr -f CGatePro-IRIX-MIPS-version.tardist
```
The CommuniGate Pro software will be installed in the `/opt/CommuniGate` directory.
- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- If your system was running sendmail, use the `chkconfig` to disable it:
```
        /sbin/chkconfig sendmail off
```
The CommuniGate Pro installation script writes the word `off` into the `/etc/config/sendmail` file.
- The installation script creates a startup script `/etc/init.d/CommuniGate`, and the symbolic links `/etc/rc2.d/S75CommuniGate` and `/etc/rc0.d/K05CommuniGate` for that script.
- The installation script creates the "base directory" `/var/CommuniGate` and the Server uses this directory by default. You can move the "base directory" to any other location. In this case, open the `/etc/init.d/CommuniGate` script file and update it.
- Restart the system or launch the start-up script manually:
```
        /etc/init.d/CommuniGate start
```
- Proceed with [Initial Configuration](#).

# Installing on an SCO UnixWare System

- Make sure that UnixWare 7.1 or better is installed.
- Log in as a super-user (root).
- Create a new directory, and "cd" into that directory; download the CGatePro-UnixWare-*version*.tar.gz archive.
- Unpack the CommuniGate Pro archive with the `gunzip` and `tar` commands:
```
        gunzip CGatePro-UnixWare-Intel-version.tar.gz
        tar -xpf CGatePro-UnixWare-Intel-version.tar
```
- Install the CommuniGate Pro package:
```
        pkgadd -d `pwd`
```
The CommuniGate Pro software will be installed in the `/usr/local/sbin` directory.
- If your system was running `sendmail` or any other SMTP server, stop that server and modify the OS start-up scripts so the system will not start that other SMTP server automatically.
- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- The Installer creates a symbolic link `/bin/cgmail` for the command line mode mail program to use with the CommuniGate Pro system.
- The Installer creates a startup script `/etc/init.d/STLKCGPro.init`, and the symbolic link `/etc/rc2.d/S88CommuniGate` for that script.
- The Installer creates the "base directory" `/var/CommuniGate` and the Server uses it by default. You can move the "base directory" to any other location. In this case, open the `/etc/init.d/STLKCGPro.init` file and update it.
- Restart the system or launch the start-up script manually:
```
        /etc/init.d/STLKCGPro.init start
```
- Proceed with [Initial Configuration](#).
  **Note:** UnixWare 7.1 has a very small per process limit for open listeners. To avoid the problem, the CommuniGate Pro ACAP server is disabled by default, and the LDAP server does not create a listener to accept secure connections. Do not try to create additional listeners before this limit is increased.

# Installing on an SCO OpenServer System

- Make sure that OpenServer 6.0 or better is installed.
- Log in as a super-user (root).
- Create a new directory, and "cd" into that directory; download the CGatePro-OpenServer-*version*.tar.gz archive.
- Unpack the CommuniGate Pro archive with the `gunzip` and `tar` commands:
```
        gunzip CGatePro-OpenServer-Intel-version.tar.gz
        tar -xpf CGatePro-OpenServer-Intel-version.tar
```

Install the CommuniGate Pro package:
```
pkgadd -d `pwd`
```
The CommuniGate Pro software will be installed in the `/opt` directory.

- If your system was running `sendmail` or any other SMTP server, stop that server and modify the OS start-up scripts so the system will not start that other SMTP server automatically.
- If your system was running SMTP, POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- The Installer creates a symbolic link `/bin/cgmail` for the command line mode mail program to use with the CommuniGate Pro system.
- The Installer creates a startup script `/etc/init.d/STLKCGPro.init`, and the symbolic link `/etc/rc2.d/S88CommuniGate` for that script.
- The Installer creates the "base directory" `/var/CommuniGate` and the Server uses it by default. You can move the "base directory" to any other location. In this case, open the `/etc/init.d/STLKCGPro.init` file and update it.
- Restart the system or launch the start-up script manually:
  ```
  /etc/init.d/STLKCGPro.init start
  ```
- Proceed with [Initial Configuration](#).

---

# Installing on an IBM OS/400 System

- Make sure you have OS/400 version V5R3 or later.
- Decide where to place the CommuniGate Pro *base directory*:
  - The files used by the CommuniGate Pro Software must reside in a thread-safe file system. Only the following file systems are thread-safe:

    `/` (root), `QOpenSys`, `QNTC`, `QSYS.LIB`, `QOPT`, `QLANSrv`, user-defined.

    For details see the Integrated File System concepts book in the [AS/400 Information Center](#).
  - If you need to have case-sensitive Mailbox names you should place the CommuniGate Pro *base directory* into a case-sensitive file system. The `QOpenSys` file system is case-sensitive. User-defined file systems can be created as case-sensitive, too. The Installer program allows you to create a case-sensitive file system for the CommuniGate Pro *base directory*.
    **Note:** If you are upgrading the CommuniGate Pro Server and the *base directory* already exists, then installer will ignore the case-sensitivity option you have specified, and will leave the *base directory* unmodified.
- If your OS/400 system was running SMTP, POP, IMAP, ACAP, or poppwd servers, stop those servers, and modify the start-up defaults, so the system will not start those legacy servers automatically. You can change the start-up defaults using the AS400 Operations Navigator.
- Make sure your OS/400 system has its TCP/IP stack active and the FTP server running.
- Download the current CommuniGate Pro OS400 version (the CGatePro-OS400-AS400.exe file) to a computer running MS Windows OS and connected (via a TCP/IP network) to your OS/400 system.
- Run the `CGatePro-OS400-AS400.exe` installer program on that MS Windows system. Follow the instructions the installer program provides.
- On the OS/400 system, start the CommuniGate Pro Server job: `QGPL/STRCGSRV`. The CGSERVER job will start in the `QSYSWRK` subsystem.
- You may want to create an autostart job entry for the CommuniGate Pro Server, or run it in a specially created subsystem. To setup the server as an autostart job:

  Add request data to the server's job description:
  ```
  CHGJOBD JOBD(CGSERVER/CGSERVERJD) RQSDTA('QGPL/STRCGSRV')
  ```
  Add an autostart job entry:
  ```
  ADDAJE SBSD(QSYSWRK) JOB(CGSERVER) JOBD(CGSERVER/CGSERVERJD)
  ```

  If you have chosen the *case-sensitive Mailbox names* option, then additional actions are required:

Authorize CGATEPRO to the command MOUNT:

```
GRTOBJAUT OBJ(CGATEPRO) OBJTYPE(*USRPRF) USER(CGATEPRO) AUT(*USE)
```

Grant CGATEPRO the *IOSYSCFG special authority:

```
CHGUSRPRF USRPRF(CGATEPRO) SPCAUT(*IOSYSCFG)
```

- **Note**: the source code of the STRCGSRV and ENDCGSRV commands is included into the CGatePro distribution set, so you can modify those commands to meet your needs.
- Proceed with [Initial Configuration](#).

---

# Installing on an OpenVMS System

- Make sure you are running OpenVMS version 7.2 or better.
- Log in as the SYSTEM user.
- If you plan to use the Server for a medium or heavy load, make sure that your system has at least 2048 "permanent I/O channels":

```
MCR SYSGEN SHOW CHANNELCNT
```

- Download the CGatePro-OpenVMS-*platform-version*.zip archive file.
- Unzip the archive file to get a POLYCENTER package file: STLK-*platform*-CGATEPRO-V*version*-1.PCSI
- Install the CommuniGate Pro package:

```
PRODUCT INSTALL CGatePro
```

The CommuniGate Pro software will be installed in the SYS$COMMON:[CommuniGate] directory.
The CommuniGate Pro startup file STARTUP.COM can be found in that directory.
- If your system was running any other SMTP, POP, IMAP server software, stop those packages and modify the OS configuration so the system will not start those other server programs automatically. Use the TCPIP$CONFIG command to disable the SMTP, POP, and IMAP servers that come with the OS.
- The Installer creates the "base directory" SYS$SPECIFIC:[CommuniGate] and the Server uses it by default.
- You may want to add the

```
$ @SYS$COMMON:[CommuniGate]STARTUP.COM START
```

command to your SYS$MANAGER:SYSTARTUP_VMS.COM file to start CommuniGate Pro automatically on system startup.
You may want to add the

```
$ @SYS$COMMON:[CommuniGate]STARTUP.COM STOP
```

command to your SYS$MANAGER:SYSHUTDWN.COM file to shut down the CommuniGate Pro server before the system shuts down.
- Proceed with [Initial Configuration](#).

---

# Installing on a QNX System

- Make sure you are running QNX version 6.2 or better.
- Log in as a super-user (root).
- Download the CGatePro-QNX-*platform-version*.qpr package.
- Install the CommuniGate Pro package:

```
cl-installer -i CGatePro-QNX-platform-version.qpr
```

The CommuniGate Pro software will be installed in the /opt directory.
- If your system was running sendmail or any other SMTP server, stop that server and modify the OS start-up

scripts so the system will not start that other SMTP server automatically.

- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- The Installer adds a reference to the CommuniGate Pro startup script to the `/etc/rc.local file`, so the CommuniGate Pro Server is started automatically when the QNX system starts.
- The Installer creates the "base directory" `/var/CommuniGate` and the Server uses it by default. You can move the "base directory" to any other location. In this case, open the start-up script file (`/opt/CommuniGate/Startup.sh`) and update it.
- Restart the system or launch the start-up script manually:

```
/opt/CommuniGate/Startup.sh start
```

- Proceed with [Initial Configuration](#).

---

# Installing on an IBM OS/2 System

- Make sure you are running OS/2 version 2.4 or better.
- Make sure you are running OS/2 TCP version 4.1 or better.
- Download the `CGatePro-OS2-platform-version.zip` package.
- Install the CommuniGate Pro package:

```
cd C:\

unzip CGatePro-OS2-platform-version.zip
```

The CommuniGate Pro software will be installed in the `C:\STALKER\CommuniGate` directory.
- If your system was running `sendmail` or any other SMTP server, stop and disable that server.
- If your system was running POP, IMAP, or poppwd servers, stop and disable these servers.
- The installed software contains the `C:\STALKER\CommuniGate\Startup.CMD` batch file.
- If you do not have the STARTUP.CMD file in the root directory of your starup disk, create such a file.
- Add the following line to the STARTUP.CMD file (stored in the root directory of your startup disk):

```
CMD /C C:\STALKER\CommuniGate\Startup.CMD start
```

this line will ensure that CommuniGate Pro server is started automatically when the OS/2 system starts.
- The default "base directory" is `C:\CommuniGate`. You can move the "base directory" to any other location. In this case, open the start-up script file (`C:\STALKER\CommuniGate\Startup.CMD`) and update it.
- Restart the system or launch the start-up script manually:

```
CMD /C C:\STALKER\CommuniGate\Startup.CMD start
```

- Proceed with [Initial Configuration](#).

---

# Installing on a MacOS Rhapsody System

- Log in as a super-user (root).
- Unpack the CommuniGate Pro archive using any uncompressing utility, or use the shell (Terminal.app) `gnutar` command:

```
gnutar xzpf CGatePro-Rhapsody-version.tgz
```

the CommuniGate.pkg package directory will be created in the current directory.
- Install the software by double-clicking the CommuniGate.pkg icon. The CommuniGate Pro software will be installed in the `/Local/Servers` directory.
- Note: The installer will create a file `/etc/startup/1950_CommuniGate`, so the CommuniGate Pro Server will auto-start when the System starts.

- Note: The installer will disable the file `/etc/startup/1800_Mail` (`/etc/startup/1900_Mail` on DR Rhapsody versions) used to auto-start `sendmail` on your system.
- If your system was running POP, IMAP, or poppwd servers, remove the lines describing those servers from the `/etc/inetd.conf` file.
- The Installer renames the `/usr/bin/mail` program into the `/usr/bin/LegacyMail`. If you decide to uninstall the CommuniGate Pro system, the legacy mail program will be renamed back to `/usr/bin/mail`.
- The Installer creates the new `/usr/bin/mail` application - a drop-in substitution for the legacy `mail` program.
- The Installer creates the "base directory" `/Local/CommuniGate` and the Server uses it by default. You can move the "base directory" to any other location. In this case, open the `/etc/startup/1950_CommuniGate` file and update it.
- Restart the system or launch the startup script manually:
    `/etc/startup/1950_CommuniGate`
- Proceed with [Initial Configuration](#).

## Installing on a BeOS System

- Make sure that BeOS R5 or better is installed.
- Download the CommuniGate Pro package: `CGatePro-BeOS-version.pkg`.
- Double-click the package file and install it.
  The CommuniGate Pro software will be installed in the `/boot/Servers/` directory.
- If your system was running `sendmail` or any other SMTP server, stop that server and modify the OS start-up scripts so the system will not start that other SMTP server automatically.
- If your system was running POP, IMAP, or poppwd servers controlled with the `inetd` daemon, remove the lines describing those servers from the `/etc/inetd.conf` file.
- The Installer adds commands to the `/boot/home/config/boot/UserBootscript` and `/boot/home/config/boot/UserShutdownFinishScript` files. Those commands call the CommuniGate Pro startup script when the BeOS starts and shuts down.
- The Installer creates the "base directory" `/var/CommuniGate` and the Server uses it by default. You can move the "base directory" to any other location. In this case, open the `/boot/Servers/CommuniGate/Startup` file and update it.
- Restart the system or launch the start-up script manually:
    `/boot/Servers/CommuniGate/Startup start`
- Proceed with [Initial Configuration](#).

**Note:** Under BeOS, each thread in a multi-threaded application is seen as a "process" when you use the `ps` system utility. As a result, you may see 30+ `CGServer` "processes" when the server is just started, and more after it has been actively used. All those "processes" are actually CommuniGate Pro Server threads, and they share all their resources - VRAM, File Descriptors, etc.

## Initial Configuration

When the CommuniGate Pro software is installed:

- Restart the OS or start the CommuniGate Pro Server manually.
- **Within 10 minutes**, connect to the WebAdmin Interface with any Web browser, using the port number 8010. Type the following URL in your browser:

      `http://your.server.domain:8010`

  where *your.server.domain* is the domain name or the IP address of the computer running the CommuniGate Pro Server.
- Read the License Agreement, enter the preferred password for the `postmaster` Account, and click the Accept

button.
- You will be redirected to the QuickStart page. Use the `postmaster` name and the selected password to open the page.
- Proceed using the [Quick Start](#) instructions.

If you failed to enter a new postmaster password within 10 minutes, the Server will be shut down. When you are ready to enter the password, repeat the steps listed above.

The [Migration](#) section can help you to schedule your CommuniGate Pro deployment process.

---

# Upgrading to a Newer Version

When you upgrade to a new version, the files in the *application directory* are substituted with the new files.

The *base directory* and all its files are not modified when you upgrade the CommuniGate Pro Server software, so all Accounts, Mailboxes, messages, settings, File Storage files, Licenses, customized Web Skin, and Real-Time Application files stay in place and continue to work with the new version of the CommuniGate Pro software.

**Note**: if you chose to manually modify Web Skin or Real-Time Application files right in the application directory, then you should save them before upgrading.

To upgrade:

- Download the new version of the CommuniGate Pro Software.
- Stop the CommuniGate Pro Server.
- Remove the previous version of the software using the same installation utility you used to install the software (the *base directory* will not be removed). This step is needed if the OS installer does not allow you to install a new version "on top" of the old one (Solaris, FreeBSD, Linux).
  **Note:** If you are using the Linux `rpm` package manager, do not use its "update" option: uninstall the old version, then install the new one:

        rpm -e CGatePro-Linux
        rpm -i CGatePro-Linux-version.rpm

- Install the new version of the CommuniGate Pro Software.
- Start the CommuniGate Pro Server.

---

# Moving to a New Hardware Server

You may want to move your CommuniGate Pro server to a different computer - running the same or a different OS. All your module settings, Account and Domain settings, Mailboxes, licenses, and other data can be preserved.

CommuniGate Pro keeps all its data in the [base directory](#). This is the only directory you need to copy to your new server computer.

CommuniGate Pro uses the same file formats on all hardware and software (OS) platforms, so usually you can just pack the entire CommuniGate Pro base directory into an archive file (using `tar` and `gzip` on Unix systems, `zip` on MS Windows systems), and unpack the archive on the new server computer.

Additional processing is needed when you move the CommuniGate Pro Server from a computer running any MS Windows operating system to a computer running any flavor of Unix, or vice versa. CommuniGate Pro files are text files, and text files on MS Windows and Unix have different EOL (end of line) symbols: CR-LF (return - linefeed) on MS Windows and bare LF (linefeed) symbol on Unix systems. To copy files properly, you may want to use any FTP software to copy files between those systems: when an FTP client is instructed to transfer files in the ASCII mode,

it properly converts EOL symbols.

**Note:** CommuniGate Pro base directory can contain non-text (binary) files in the `WebSkins` and `PBXApps` directories inside the Account and Domains subdirectories - graphic, audio, and video files used in the custom Skin Interfaces and Real-Time Applications.
Account File Storage can also contain binary files. These files are stored in the `account.web` directories inside the Account directories.
When you move a CommuniGate Pro *base directory* between systems using different EOL conventions, check that those binary files are copied in the BINARY mode (i.e. without EOL re-coding).

If the new server computer is running a Unix system, check that the copied directory and all its files and subdirectories have the same access rights as they had on the old system.

After the CommuniGate Pro *base directory* is copied, download and install the proper version of the CommuniGate Pro Server on the new server computer. There is no need to copy the content of application directory from the old server computer, even if both new and old computers are running the same operating system.

Check that the newly installed copy of the CommuniGate Pro Server (its startup script, if any) is configured to use the copied base directory, and then start the CommuniGate Pro Server on the new computer. Use the WebAdmin Interface to modify the computer-related settings on the new server. For example, you may need to update the Client IP Addresses table or re-assign IP addresses to CommuniGate Pro Domains.

**CommuniGate Pro**

| Installation | **QuickStart** | Migration |
|---|---|---|

# Quick Start

- **Main Domain Name**
- **Language and Character Set**
- **Time Zone**
- **Network**
- **Accounts**

When you have your CommuniGate Pro Server installed and the `postmaster` password has been created, you need to configure the Server before you can start using it.

This section outlines the basic CommuniGate Pro configuration options and provides references to in-depth information.

If you have any problem with your configuration, please check the How To and Help Me sections.

# Main Domain Name

Specify the proper name in the Main Domain Name field and click the Update button. The name should be the full name (qualified name) of your company Domain. [read more]

If you need to serve (to *host*) several domains on your CommuniGate Pro Server, you may want to create additional (*secondary*) Domains: each Domain is independent and it has its own set of Accounts, Settings, and other objects [read more]

# Language and Character Set

The CommuniGate Pro default Language is English. If most of your users prefer a different language, use the Language menu to select the preferred one. You may also want to change the Default preferred character set. While most of modern interfaces use the universal Unicode (UTF-8) character set, preferred character sets are used when reading E-mails sent with an unspecified character set, and when sending E-mails to legacy E-mail systems.

These settings will be set in the Server-wide Account Default Preferences. [read more]

# Time Zone

Check the displayed Server Time. First, make sure that the Time Zone (*+nnnn* or *-nnnn*) is the correct one. Then, check the displayed Server Time.

If any of these parameters are incorrect, fix your Server OS settings. You may have to restart the OS and/or the CommuniGate Pro Server to active the new OS time settings.

If the Server Time is specified correctly, select the Time Zone most of your user will use.
If you select the "built-in" zone (HostOS), the Server will use a fictitious zone that has the same time difference with GMT as the Server OS has at this time. This zone has no support for daylight saving time and it cannot be used for sending recurrent events outside your Server. Unless your Time Zone is not listed, avoid selecting the "built-in" zone.
[read more]

# Network

Open the Settings WebAdmin Realm and open the Network section. If your Server is connected to your office or home LAN, specify the LAN network addresses. [read more]

# Accounts

Open the Domains WebAdmin Realm and create Accounts for your users. [read more]

| Installation | QuickStart | **Migration** |
|---|---|---|

# Migrating to CommuniGate Pro

- **Supporting Network Users**
- **Supporting Local Users**
- **Using Legacy Mailboxes**
- **Converting Passwords**
- **Migrating from** `sendmail`
- **Migrating from** `Post.Office®` **Servers**
- **Migrating from** `Netscape®/iPlanet Messaging` **Servers**
- **Migrating from** `IMail®` **Servers**
- **Migrating from CommuniGate/MacOS and SIMS**
- **Migrating from Microsoft® Exchange Servers**
- **Copying Mailboxes from Other POP Servers**
- **Copying Mailboxes from Other IMAP Servers**
- **Copying All Mailboxes from Other Servers**
- **Migrating from an Arbitrary Server ("on-the-fly" migration)**
- **Switching Servers**
- **Moving To Secondary Domains**

If your system was already running some type of E-mail server, you may want to integrate your existing E-mail environment into the CommuniGate Pro messaging system.

To migrate all your users, you need:

- to create all existing mail accounts on your new CommuniGate Pro Server, using already specified account settings, especially - account passwords
- to migrate all user E-mail from the old server to the new CommuniGate Pro Server
- provide services for "local users" - the users of "local" (Unix) mail programs that directly access their mailbox files, bypassing E-mail protocols (such as the POP or IMAP protocols).

# Supporting Network Users

Users that access their mail accounts using any standard Internet Protocol (POP, IMAP) do not have to switch their mailer applications or change mailer settings - CommuniGate Pro supports not only the published mail access protocols standards, but most of unofficial protocol extensions, too.

# Supporting Local Users

Some users registered with the server OS may access their mail accounts using legacy mailer applications that read mailbox files directly. Since these mailers bypass Internet protocols, the CommuniGate Pro server has no control over those mailers.

The CommuniGate Pro Server keeps all user Accounts and Mailboxes inside its "base directory". On a properly

configured system direct user access to the base directory is prohibited to ensure that Account Mailboxes and other Server files stay intact.

When you create a CommuniGate Pro Account for a local user who needs mail access via legacy mailers, select the Legacy INBOX option. In this case, the INBOX Mailbox will not be created inside the CommuniGate Pro *base directory*. Instead, the INBOX Mailbox location will be taken from the Domain settings.
Usually, you specify `/var/mail/*` or `/var/spool/mail/*` - the "standard" location where legacy mailers expect to see user Mailboxes.

When the Server accesses these *Legacy Mailboxes*, it uses the OS file-locking mechanism to synchronize its activity with legacy mailers.

**Note:** legacy mailers were not designed to support simultaneous access to mailboxes. They can destroy data in your mailbox if you open two legacy mailer sessions with the same mailbox and delete messages in one of the sessions. CommuniGate Pro cannot fix this, since these mailers bypass the server, it can only guarantee (using file locks) that legacy mailers do not destroy a mailbox while CommuniGate Pro is working with it.

For more details, see the Mailboxes section.

---

# Using Legacy Mailboxes

The default format for CommuniGate Pro Mailboxes is the BSD-type text mailbox format: a Mailbox is a text file with messages separated with an empty line and a line starting with the symbols `From` .

Since most legacy mail systems use this format, the existing mailbox files can be used when migrating to CommuniGate Pro. You should either copy the old mailbox files into the proper places in the CommuniGate Pro Account directories, or you can specify that some Accounts have Legacy INBOXes (see above), and the old mailbox files will be used "in place".

**Note:** when CommuniGate Pro stores a message in a BSD-type Mailbox, it adds additional fields to the separator (`From`) line. These fields are ignored by legacy mailers and mail servers, but they allow the CommuniGate Pro Server to keep the message status and unique message ID information. When you make your CommuniGate Pro Server use a BSD-type mailbox file composed with an old mail system, it issues warnings (Log records) about missing fields in the message separators. The Server still opens such mailboxes: it creates empty status flag sets for such messages, and it generates unique IDs on-fly. As users read, move, and/or delete old mail from their mailboxes, messages stored with the old mail system disappear, and the Server stops complaining when opening these mailbox files.

---

# Converting Passwords

If your old mail server authenticated clients are using the Unix OS account and passwords (the `passwd` and `shadow` files), you can simply enable the Use OS Password option for those accounts and the CommuniGate Pro Server will use the OS authentication procedures for them.

Since the OS Passwords are one-way encrypted, they cannot be used for secure SASL authentication methods. The following procedure can be used to allow migrated users to employ the secure SASL methods:

- enable the Use OS Password option either in the Default Account Settings or in the Account Template.
- specify an empty string for the CommuniGate Password in the Account Template.
- import all accounts without the Password field.

Users can connect to the newly created accounts using their old OS Passwords - i.e. the same passwords they used

with the legacy mail system. When users try to modify their account passwords, the new passwords will be stored as CommuniGate Passwords. All users that have updated their passwords will be able to use the secure SASL authentication methods.

Sometimes you cannot use this method. For example, you migrate users from a different server, and you do not register them all with the Unix OS on the new server, but you do have the `passwd` file from the old server. In this case, you may want to enter the Unix-style (`crypt`-encrypted) passwords as the CommuniGate (internal) Passwords.

To make the CommuniGate Pro server process its internal password string as a `U-crpt` (`crypt`-encrypted) or other support encrypted password (see below), store the password in the Account Settings with one-byte binary 002 prefix. If you want to create a user `test` using the [CLI](CLI) interface, and the Unix (`crypt`-encrypted) password for that user is `AslUzTlJkPsocc`, then use the following CLI command:
```
createaccount "test" {Password="\002AslUzTlJkPsocc";}
```

If you create users by [importing](importing) an account list from a text file, place the Unix passwords strings into the `UnixPassword` column, not into the Password column. In this case, the Loader will automatically add the binary 002 prefix to all password strings. If you create users using the [LDAP Provisioning](LDAP Provisioning) feature, specify the encrypted password as the `unixPassword` attribute.

Account Settings of the new accounts should specify one of the CommuniGate Pro *password encryption* methods (clear text or A-crpt). Users will be able to log in using their old Unix/encrypted passwords. When they update their passwords, new CommuniGate Password strings will be stored using the specified CommuniGate Pro password encryption method. All users that have updated their passwords will be able to use the secure SASL authentication methods.

Some servers produced by the Netscape and Software.com companies store user passwords using several encryption methods. When passwords are retrieved from those servers, they have the following form:
*{method}eNcoDeD*
or
*$method$eNcoDeD*
where *method* specifies one of the standard encryption methods, and the *eNcoDeD* string is an encrypted password (sometime - Base64-encoded).

CommuniGate Pro can use these passwords in the same way it uses the Unix-encrypted passwords, and they should be entered in the same way: you should use the binary 002 prefix in the CLI commands and/or you should place those passwords into the UnixPassword field of the [Account Import file](Account Import file).

The following encryption methods are supported:

- {crypt} - the standard Unix crypt method.
- {WM-CRY} - the standard Unix crypt method (same as {crypt}).
- {MD5} - the MD5 digesting method (Base-64 encoded MD5 digest of the password string).
- {SHA} - the SHA1 digesting method (Base-64 encoded SHA1 digest of the password string).
- {NS-MTA-MD5} - the MD5-based method used in the Post.Office servers and old Netscape Messaging Servers (the eNcoDeD portion contains 64 hexadecimal digits).
- {SSHA} - the "salted SHA1" digesting method (Base-64 encoded SHA1 digest of the password and salt strings, followed with the salt bytes). This method is used in the Sun Directory Server application.
- {LANM} - the "LAN Manager" hash used in Microsoft Windows servers (the eNcoDeD portion contains 32 hexadecimal digits).
- {MSNT} - the "Microsoft NT" hash used in Microsoft Windows servers (the eNcoDeD portion contains 32 hexadecimal digits).
- $1$ - MD5-based password hash used in FreeBSD and some other Unix systems.
- $2a$ - BlowFish-based password hash used in OpenBSD and some other Unix systems.

The following is a sample Import file:

| Name | UnixPassword | Password Type |
|------|--------------|---------------|
| user1 | YIdhkjeHDKbYsji | Unix-crypt |
| user2 | {SHA}Ue4Erbim2TC7CmuukMOBejeytr2= | SHA1-digested |
| user3 | {MD5}zverMUhsgJUIDjeytr2= | MD5-digested |
| user4 | {crypt}YIdhkjeHDKbYsji | Unix-crypt, same as for the `user1` account |
| user5 | $1$VlPrB$vNjOAytB3W.j0bkkbaN2Z. | BSD-type MD5-encrypted |

You can use a CommuniGate Pro CLI script to automatically import all users and their passwords from the OS `/etc/passwd` file. See the [CommuniGate Perl Interface](#) site for a sample script.

---

# Migrating from `sendmail`

If you are migrating from a `sendmail`-based mail system, you may find the following information useful:

The `aliases` file
> The sendmail aliases file allows the administrator to redirect local mail to one or several addresses. Sendmail uses the term "alias" for too many different things, so you should select the proper CommuniGate Pro feature to implement different types of sendmail "aliases":
>
> * Each account can have one or several [aliases](#). Mail sent to any Account Alias name is routed to the Account itself. If the `domain.dom` Account `john.smith` has `j.smith` and `smith` aliases, mail sent to `j.smith@domain.dom` and `smith@domain.dom` addresses is delivered to the `john.smith@domain.dom` Account. When an Account is renamed, its Aliases automatically start to point to the new name, and when an Account is removed, all its Aliases are removed, too.
> * Domain [Forwarder](#) objects allow you to redirect mail sent to a Domain address to any other address. The `domain.dom` Forwarder `susan.smith` can reroute all mail sent to `susan.smith@domain.dom` address to the `susan@otherisp.dom` address.
> * Domain [Group](#) objects allow you to redirect mail sent to some Domain address to any set of addresses.
> * The [Router](#) module allows you to redirect mail sent to a certain address to any other address. The Router Alias Record `<*.smith@domain.dom> = Smith@domain.dom` will reroute mail sent to `john.smith@domain.dom` and to `susan.smith@domain.dom` to the `Smith@domain.dom` address.
> * The [Account Rules](#) allow the administrator and/or the users themselves to redirect/forward/mirror all or certain mail to one or several addresses.
> * The [Server-Wide Rules](#) allow the administrator to redirect/forward/mirror all or certain mail to one or several addresses.
> * The shared or [Foreign Mailboxes](#) feature allows a user to grant access to a Mailbox to other users; in many cases a shared Mailbox is a much better alternative to mail distribution.
> * The [LIST](#) module provides a very powerful mailing list distribution mechanism.

`procmail` processing
> The Server-Wide, Domain-Wide, and Account-Level [Automated Rules](#) allow administrators and users to perform automatic mail processing and filtering using the powerful condition checking and processing operations built into the CommuniGate Pro Server.
> For the situations where messages should be processed using an external filter or processor, the `Execute` Automated Rules operation can be used to start the specified external program as a separate OS task (for example, the Rules can be used to process all or certain incoming messages with the same `procmail` program).

---

# Migrating from `Post.Office®` servers

The Post.Office product stores account names, passwords, and other information in its account database. The special [Post.Office Migration Utility](#) can be used to retrieve that information and to store it in a tab-delimited file that can be used with the CommuniGate Pro WebAdmin [Account Import](#) function.

The [List Migration](#) script allows you to copy mailing lists and their subscriber sets from Post.Office to CommuniGate Pro.

---

# Migrating from `Netscape®/iPlanet Messaging` servers

The Netscape (iPlanet) Messaging server stores account names, passwords, and other information in a Directory server "subtree". Use regular LDAP tools to export the Directory subtree into an LDIF file. The special [Netscape Migration Script](#) can be used to convert the account information from the LDIF format into a tab-delimited file that can be used with the CommuniGate Pro WebAdmin [Account Import](#) function.

---

# Migrating from `IMail®` servers

The IMail product stores account names, passwords, and other information in its account database. The special [IMail Migration Utility](#) can be used to retrieve that information and to store it in a tab-delimited file that can be used with the CommuniGate Pro WebAdmin [Account Import](#) function.

---

# Migrating from CommuniGate/MacOS and SIMS

If you want to move your users from a CommuniGate for MacOS server, you can build the account list file using the [CommuniGate/MacOS extractor](#) utility.

If you want to move your users from a Stalker Internet Mail Server (SIMS), you can build the account list file using the [SIMS extractor](#) utility.

---

# Migrating from Microsoft® Exchange Servers

The special [Exchange Migration Utility](#) allows you to retrieve user lists from an Exchange server and to create those users in CommuniGate Pro Domains. The utility copies all user folder data (mail, calendaring, contacts, etc.) converting data and address formats "on-the-fly".

The utility also extracts the Exchange Global Address Book and converts it into an LDIF file that can be imported into the CommuniGate Pro Directory.

The utility requires an MS Windows workstation to run.

---

# Copying Mailboxes from Other POP Servers

When migrating from other mail servers, you may want to copy all messages from an account on the old server to the already created account on the new server.

The CommuniGate Pro software package includes the `MovePOPMail` program. This program connects to the old POP server, logs in, retrieves all messages, and submits it to the new SMTP server:

    MovePOPMail [--verbose] [--delete] [--notimeout] *POPserver POPname POPpassword SMTPserver SMTPrecipient*

*POPserver*
The IP address of the old (source) POP3 server; if that POP server operates on a non-standard TCP port, you can specify the port number using the colon symbol: `192.0.2.3:111` - POP server at the 192.0.2.3 address, port 111.

*POPname*
The POP account name - i.e. the name of the source account on the POP server.

*POPpassword*
The POP account password.

*SMTPserver*
The IP address of the new (target) SMTP server; if that SMTP server operates on a non-standard TCP port, you can specify the port number using the colon symbol: `192.0.2.4:26` - SMTP server at the 192.0.2.4 address, port 26.

*SMTPrecipient*
The address to send the retrieved messages to. Usually - the account name on the new server.

`--verbose`
An optional parameter. When specified, the progress information is sent to the standard output.

`--delete`
An optional parameter. When specified, the program deletes all retrieved messages from the POP account.

`--notimeout`
An optional parameter. When specified, the program increases the SMTP and POP operation time-out values from 20 seconds to 1 hour.

Sample:
```
MovePOPMail --verbose 192.0.0.4 john "jps#dhj" 192.0.1.5 john
```

# Copying Mailboxes from Other IMAP Servers

When migrating from other mail servers, you may want to copy all mailboxes and all messages from an account on the old server to the already created account on the new server.

The CommuniGate Pro software package includes the `MoveIMAPMail` program. This program connects to the old and new IMAP servers, logs into both, retrieves the list of mailboxes in the old account, creates all missing mailboxes in the new account, and copies all messages from mailboxes in the old account to the mailboxes in the new account. The program also copies the list of "subscribed mailboxes", and the mailbox ACLs (if supported).

    MoveIMAPMail [*flags*] *OldServer oldName oldPassword NewServer newName newPassword*

*oldServer*
The IP address of the old (source) IMAP4 server; if that IMAP server operates on a non-standard TCP port, you can specify the port number using the colon symbol: `192.0.2.3:144` - IMAP server at the 192.0.2.3 address, port 144.

*oldName*, *oldPassword*
Strings to use when logging into the source IMAP server.

*newServer*
>The IP address of the new (target) IMAP4 server; if that IMAP server operates on a non-standard TCP port, you can specify the port number using the colon symbol: `192.0.2.5:145` - IMAP server at the 192.0.2.5 address, port 145.

*newName*, *newPassword*
>Strings to use when logging into the target IMAP server.

*Flags* is zero, one or several optional parameters:

`--verbose`
>An optional parameter. When specified, the progress information is sent to the standard output.

`--list` *search*
>An optional parameter. When specified, the following *search* string is used to find all mailboxes in the source account. Some IMAP servers show the entire user directory or even system directory if the default search string `"*"` is used. Consult with your old IMAP server manual to learn the search string to use.

`--source` *prefix*
>An optional parameter. When specified, the following *prefix* string is used as the first parameter of the "LIST" command (see above). If the mailbox names produced with the LIST command start with the specified prefix, the prefix is removed from the name when the mailbox is created on the target server.
>This feature allows you to copy a subtree of the source account mailbox tree into the "top" level of the target account mailbox tree. If the source account contains mailboxes `abc/mail1` and `abc/mail2`, then --source abc/ parameter can be used to copy just these 2 mailboxes and to create them as "mail1" and "mail2" mailboxes in the target account.
>If the source server is CommuniGate Pro, this parameter can be used to copy all mailboxes from any account, using the postmaster name and password: `--source '~username'/`
>See the --target option below.

`--target` *prefix*
>An optional parameter. When specified, the following *prefix* string is appended to all mailbox names sent to the target server. For example, if the target server is CommuniGate Pro, you can specify the `postmaster` credentials in the parameters (instead of the `username` credentials), and use the
>`--target '~username/'`
>parameter to copy mailboxes to the `username` account. This can be useful when you do not know the `username` account password.

`--skipMailbox` *mailboxName*
>An optional parameter. When specified, the mailboxName mailbox is not copied.
>This parameter can be specified more than once, to exclude several mailboxes.

`--notimeout`
>An optional parameter. When specified, the program increases the IMAP operation time-out value from 20 seconds to 1 hour. You may want to specify this option when your copy mail from slow servers.

`--delete`
>An optional parameter. When specified, the program deletes the retrieved messages from the source account.

`--nosubscription`
>An optional parameter. When specified, the program does not copy the mailbox subscription list to the target account.

`--subscribed`
>An optional parameter. When specified, the program copies only those mailboxes that are listed in the source account mailbox subscription list.

`--fetchRFC822`

An optional parameter. When specified, the program uses the RFC822.PEEK attribute instead of the BODY.PEEK[] attribute when it sends IMAP FETCH commands to the source server. Use this attribute when the source server is too old and does not support the BODY.PEEK[] FETCH attribute.

**--byOne**
An optional parameter. When specified, the program fetches messages from the source IMAP mailbox one by one; otherwise it fetches all messages at once. Use this parameter when the source server fails to retrieve all mailbox messages with a single FETCH command.

**--noACL**
An optional parameter.When specified, the program does not copy the mailbox ACL (access control lists) to the target account.

**--copyMailboxClass**
An optional parameter. Can be specified if both source and target servers are CommuniGate Pro servers. When specified, the program copies the mailbox classes ("calendar", "contacts", etc.)

**--fixLongLines** *number*
An optional parameter. Can be specified if the source server has messages with extremely long text lines. These lines will be separated into several lines so no message line in a target server mailbox is longer than *number* bytes.

**--proxyAuth** *username*
An optional parameter. When specified, the 'PROXYAUTH username' command is used with the source IMAP server. Use this command to login to the source server as admin to retrieve mail from an account whose password you do not know, if your source server supports the PROXYAUTH IMAP command.

**--renameMailbox** *oldMailboxName newMailboxName*
An optional parameter. When specified, the oldMailboxName mailbox name in the source account is automatically translated into newMailboxName name in the target account.
This parameter can be specified more than once, to rename several mailboxes.

**--renameMailboxList** *file.txt*
An optional parameter. When specified, a list of mailboxes renaming pairs is read from the specified text file. Each file line should contain pairs of an old and new mailbox names separated with the tabulation symbol. When a source account mailbox has a name listed as an "old" name in the file, the mailbox messages are copied into the target account mailbox with the name specified as "new" on the same file line.

**--filter [before | after ]** "*dd-mmm-yyyy*[ *hh:mm:ss*]"
An optional parameter. When specified, the program copies only messages with the INTERNALDATE before or after the specified date. The hh:mm:ss part is optional, if it is not specified `00:00:00` is assumed.

Sample:
```
MoveIMAPMail --list "Mail/*" 192.0.0.4 john "jps#dhj" 192.0.1.5 johnNew dummy
```

Note: if a mailbox name in the source account ends with symbols `.mbox` or `.mdir`, the mailbox name in the target account will end with the `-mbox` or `-mdir` symbols.

Note: if a mailbox name in the source account starts with the symbol `.` or `~`, the mailbox name in the target account will start with the `_` symbol.
Leading and trailing spaces, the `\` and `#` symbols in the source account mailbox names are replaced with the `_` symbols when the target account mailbox names are composed.

# Copying All Mailboxes from Other Servers

After you have created the accounts on your new CommuniGate Pro Server, you may want to copy mail from all mailboxes on the old server to the accounts on the new server.

The CommuniGate Pro software package includes the `MoveAccounts` program. This program uses a tab-delimited text file that contains account names and passwords. It can be the same file you have used to [Import Accounts](#) to a CommuniGate Pro domain.

The program scans the file and uses either the [MovePOPMail](#) or the [MoveIMAPMail](#) program to move messages for each account. These programs should be located in your current directory.

> MoveAccounts [--POP | --IMAP] *file sourceServer targetServer* [*suppl_parameters*]

`--POP, --IMAP`
> Parameters that specify whether `MovePOPMail` or `MoveIMAPMail` program should be used. If none is specified, the `MovePOPMail` program is used.

*file*
> The name of a tab-delimited file that contains account names and passwords.

*sourceServer*
> The IP address of the old (source) server (POP or IMAP); can include the port number.

*targetServer*
> The IP address of the new (target) server (SMTP or IMAP); can include the port number.

*suppl_parameters*
> Optional parameters (such as `--verbose`, `--delete`, `--notimeout`, `--list` *search*, etc.) passed to the `MovePOPMail` or `MoveIMAPMail` program.

The first line of the *file* should contain the data field names. The fields with names `Name` and `Password` must be present.

If the field `NewName` exists, it is used as the *SMTPrecipient* parameter when the `MovePOPMail` program is started, or as the *newName* parameter for the `MoveIMAPMail` program. Otherwise the same `Name` field data is used.

If the `--IMAP` parameter is specified, the program checks if the NewPassword field exists. If it does, the data in that field is passed as the newPassword parameter to the `MoveIMAPMail` program. Otherwise, the same `Password` field data is used.

All fields with other names are ignored.

The following is a sample AccountList file:

```
Name                    Limit               Password
john                    10K                 j27ss#45
jim                     120K                dud-ee
george                  31M                 mia#hj!
```

```
MoveAccounts --POP AccountList 192.0.0.3 192.0.1.5
```

If you cannot obtain the clear-text passwords for all accounts you want to copy, and the old server is using the Unix `/etc/passwd` or `/etc/shadow` file, follow these steps:

- Find the OS file that contains the encrypted user passwords: `/etc/passwd`, `/etc/shadow`, or `/etc/master.passwd` (see your OS documentation). We will refer to that file as `/etc/shadow`.
- Create a backup copy of the `/etc/shadow` file.
- Find the `/etc/shadow` record that contains information for your own account or any other account you know the

clear-text password for. Let us say that this known unencrypted OS account password is `mypassword`, and the encrypted password you see in the `/etc/shadow` account record is `FU3jjF/gkJJdA`.

- Edit the `/etc/shadow` file so all account records will contain `FU3jjF/gkJJdA` in their encrypted password fields.
- Open the CommuniGate Pro Default Account Settings page for the domain you are migrating. Enable the `Use OS Passwords` option and check that the `OS UserName` option is set to `*`.
- Create the AccountList file that contains the account names in the `Name` field, and the string `mypassword` in the `Password` field.
- Copy all account mailboxes from the old server to the new server using the `MoveAccounts` command. The command will successfully log into both servers, since all accounts on both servers accept the string `mypassword` as the account password.
- Restore the `/etc/shadow` from the backup copy.
- Disable the Use OS Password setting in the CommuniGate Pro Default Account Settings, if you do not want to use OS Passwords for your CommuniGate Pro Accounts.

---

# Migrating from an Arbitrary Server ("on-the-fly" migration)

Use this alternative migration method when the password switching method explained above cannot be used, and/or the user names and passwords cannot be retrieved from the old server.

The method is based on the External Authentication feature of the CommuniGate Pro server.

Download, install, and tailor the migration script, and configure CommuniGate Pro to use this script as the CommuniGate Pro External Authentication program.

Create the target CommuniGate Pro Domain, and enable the Consult External for Unknown Domain settings. Disable the Use CommuniGate Password option and enable the Use External Password option in the Account Template.

When a user attempts to connect to a non-existent Account, or when CommuniGate Pro receives a message for non-existent Account, the External Authentication script is called. The script connects to the old server using the SMTP protocol and checks if the account with the same name (same address) exists on the old server. If the old server does not reject the address, the script creates the Account with this name in the CommuniGate Pro Domain. Then the CommuniGate Pro Server delivers the message to this newly created Account.

When a user connects to the CommuniGate Pro Server, the user mailer sends the user name and the user password in the plain text form. Because the CommuniGate Pro Account has the Use CommuniGate Password option disabled, and the Use External Password option enabled, the External Authentication script is called. The script connects to the old server using the POP or IMAP protocol and checks if it can log into the old server with the provided user credentials.

If the old server accepts the specified password:

- the script uses the CommuniGate Pro CLI:
  - to set the specified password as the CommuniGate Password for this Account,
  - to switch off the Use External Password option for this Account,
  - to switch on the Use CommuniGate Password option for this Account.
- the script starts the MoveIMAPMail or MovePOPMail programs to copy the account mailboxes from the old server to the CommuniGate Pro server, or saves the name/password pairs into a text file which you can later use with the MoveAccounts program (this is a configurable option).

After the first successful login, the correct password will be set as the new CommuniGate Password, and all Account mail will be copied from the old server.

After all old server users have successfully connected to the CommuniGate Pro server at least once, all their Accounts will be created and have the correct CommuniGate Passwords set. Then you can disable the External Authentication

script and retire the old server.

# Switching Servers

Very often it is essential to switch to the new server without any interruption in the services you provide.

If you install the new CommuniGate Pro server on the same system as the old mail server, you should:

- switch CommuniGate Pro SMTP receiving to port 26, so it will not interfere with your old server SMTP activity.
- switch CommuniGate Pro POP service to port 111, and IMAP service to port 144, so they will not interfere with your old server POP/IMAP activity.
- Configure CommuniGate Pro and create Domain Aliases and/or full featured Secondary Domains.
- Create some test accounts in the main and secondary domains and check that you can log into those accounts using WebUser Interface.
- Check that you can send mail from those accounts using the WebUser Interface: mail to other test accounts in the created domains and mail to other servers should be delivered correctly.
- If you have a POP or IMAP client that allows you to specify a non-standard port number, check that you can log into the test accounts on the POP port 111 and IMAP port 144.
- Create tab-delimited file(s) with names, passwords, and other attributes of your existing accounts and use the Account Import feature to create all accounts on your new server.

All this can be done while your old server is still active.

Now, you should stop your old server activity by either changing its port numbers to non-standard values, or by disconnecting it from the external network.

Use the `AccountMove` program to copy all messages from your old server to CommuniGate Pro.

When all messages are copied, change the CommuniGate Pro SMTP port number back to 25, POP port number - to 110, and IMAP port number to 143. Now CommuniGate Pro will operate as your mail server, without any interruption in the services you provide.

You may want to keep the old server running for several hours - in case its queue contained some delayed outgoing messages. Just check that the old server does not try to use the standard ports.

# Moving to Secondary Domains

If you create several Secondary domains in CommuniGate Pro, you may want to move accounts from your old server(s) to a Secondary CommuniGate Pro Domain, not to its Main Domain.

In this case, you should add the NewName field to your account list file, and copy all names into that column, adding the `@domainname` string to each name.

If you use the IMAP protocol to move messages between the servers, you may use a simpler method:

- If the target domain has an IP address assigned to that domain, use that address in the mail copying programs: all non-qualified names provided on connections established with that address are interpreted as names in that domain. See the Access section for the details.
- If the target domain does not have an assigned IP address, temporarily assign the IP address of the main domain to that secondary domain. Move the messages, and remove the IP address from the list of addresses assigned to that domain.

Introduction

Installation

SysAdmin

Network

Objects

Storage

E-mail

Real-Time

Access

Services

Directory

Clusters

Applications

Miscellaneous

Licensing

WebMail

Pronto!

PBX

**Sys Admin** **Logs** **Router** **Protection** **Security** **PKI** **Intercept** **Scalability** **Alerts** **Statistics**

# System Administration

- **Realms and Access Rights**
- **Interface Types**
- **General Settings**
- **Specifying the Preferred Language**
- **Specifying the Preferred Time Zone**
- **Base Directory Structure**
- **Command Line Options**
  - **Specifying Command Line Options under Windows**
  - **Customizing Unix Startup Scripts**
  - **Customizing OpenVMS Startup Procedures**
- **Shutting Down**
- **OS syslog**
- **Urgent Notifications**
- **Server Root Privilege**
- **Domain Administration**
  - **Domains Administrators in other Domains**
  - **Domain Administrator Access Rights**
- **WebAdmin Preferences**
- **Customizing Domain WebAdmin Interface**
- **Customizing Server Prompts**

When the CommuniGate Pro Server is up and running, it can be configured, monitored, and set up using any Web browser.

By default, the HTTP module provides access to the CommuniGate Pro Server Administration pages (the WebAdmin Interface) via the TCP port number 8010. To use the WebAdmin Interface, use the `http://serveraddress:8010` URL, where *serveraddress* is the server IP address or the Server DNS name (A-record).

You can also administer the CommuniGate Pro Server via the network command-line interface.

# Realms and Access Rights

The WebAdmin Interface pages are grouped into five Realms. To access a page in any realm, you should have a CommuniGate Pro Account, and that Account should be explicitly granted access rights to that realm.

- The Settings realm contains pages that allow a Server Administrator to modify the Server kernel and module settings.
- The Users realm contains pages that allow a Server Administrator to create and remove Domains and Accounts, and to modify the Domain and Account settings.
- The Monitors realm contains pages that allow a Server Administrator to monitor server and module queues, communication channels and their states, to browse the Server Logs, and to view the Server Statistics.
  If a user is granted an access right to the Monitors realm, additional Monitor Access rights can be granted, too (rights to release and reject module queues, reconfigure the Log Manager, etc.)
- The Directory realm contains pages that allow a Server Administrator to configure the CommuniGate Pro Directory services.
- The Master realm contains the pages that allow a Server Administrator to grant and revoke Server Administrator access rights, and to modify the Server License Keys.

**Note:** An Account should be granted the Domains Read access right to access the Users realm to read information about all Domains and Accounts.

**Note:** An Account should be granted the All Domains access right to access the Users realm to read information and to make modifications in all Domains and Accounts.

**Note:** If an Account is granted the Master access right, the Account user can access all realms and make all types of modifications.

**Note:** These Server Administration access rights can be granted to the Main Domain Accounts only. Accounts in secondary Domains can be granted Domain Administration rights only.

When a Server is installed for the first time, it creates the `postmaster` Account in the Main Domain, and it grants the Master access right to that Account.

# Interface Types

The CommuniGate Pro Server is very complex and flexible software. Its set of Settings and other configuration options can be overwhelming for someone not familiar with the product.

An WebAdmin Interface Type ranging from `Basic` to `Expert` is assigned to each Administrator. These Types (or Expertise Levels) are designed to simplify the learning process:

- `Basic`: the WebAdmin Interface is shrunk to the minimal set of pages, and only the essential settings are displayed on these pages.
- `Advanced`: the WebAdmin Interface shows all pages and settings needed to control a typical installation.
- `Expert`: the WebAdmin Interface shows all available pages and settings.

When the CommuniGate Pro Server is installed for the first time, and the `postmaster` Account is created, the `Basic` WebAdmin Interface Type is set for that Account. You can change the Interface Type by opening the WebAdmin Preferences pages.

**Note:** This documentation shows the WebAdmin pages and settings as they are displayed in the `Expert` mode.

# General Settings

Use the WebAdmin Interface to open the General page in the Settings realm:

| | |
|---:|:---|
| **Main Domain Name:** | |
| **Server Internals Log:** | Low Level |
| **Crash Recovery:** | Disabled |
| **Separate WebAdmin Realms:** | No |
| **Server Time:** | Wed, 06 Dec 2006 00:38:40 -0800 |
| **Server Up-time:** | 125 day(s) 5 hour(s) 23 min 16 sec |
| **Server OS:** | Sun Solaris |
| **Server Hardware:** | x86 (32-bit) |
| **Server Version:** | 5.1.7 |
| **MAPI Version:** | 1.2.8 |
| **IPv6 Support:** | Enabled |
| **Name Server(s) IP Address(es):** | [64.173.55.167] |
| **Server IP Address(es):** | [64.173.55.171] mycompany.com<br>[64.173.55.170] client2.dom<br>[2001:470:1f01:2565::a:845] mycompany.com |
| **Startup Parameters:** | `"--logtoconsole"` |

`Main Domain Name`
> In this field you should enter the name that the CommuniGate Pro Server will interpret as its own *Main Domain Name*. All mail addressed to that domain will be treated as local, and (in the simplest case) that mail will be stored in local Account Mailboxes. Initially, this field contains the server computer name that CommuniGate Pro retrieves from the OS. If this names looks like `host12345hh.company.com`, you should change it to the name of the domain this Server should process.

**Note:** unless you create additional Domains ONLY the E-mail Messages and Signals directed to addresses in the Main Domain will be processed as *local*. If the Main Domain Name is entered as `company.com`, then Messages to `mail.company.com` or Signals to `sip.company.com` will not be processed as local, and if such a Message or Signal is received, the Server will try to deliver it to the `mail.company.com` or the `sip.company.com` system over the network.
If the DNS record for `mail.company.com` or `sip.company.com` points to the same Server computer, the *mail loop* or *signal loop* error will be detected, and the Message or Signal will be rejected.

If your Server should serve several domain names, enter the additional domain names as Main Domain Aliases (if those domain names should be "mapped" to the Main Domain), or create additional ("secondary") Domains.

Sample configuration:
> Your Server should serve the `company.com` and `client1.com` domains. These domain names have their DNS MX-records and SIP SRV-records pointing to `server.company.com` and `server.client1.com` A-records, and these A-records point to IP address(es) assigned to your CommuniGate Pro Server system.

> - set `company.com` as the Main Domain Name.
> - open the Domains page, find the **company.com** record and click on its `Settings` link to open the company.com Domain Settings page. Scroll it down to find the Domain Aliases fields.
> - enter `server.company.com` into an empty Aliases field, and click the Update button.
> - open the Domains page. Enter `client1.com` into the text field and click the Create Domain button.
> - the `client1.com` record should appear in the list; click its `Settings` link to open the client1.com Domain Settings page. Scroll it down to find the Domain Aliases fields.
> - enter `server.client1.com` into an empty Aliases field, and click the Update button.

`System Internals Log`
> Use this setting to specify what kind of information the Server kernel module should put in the Server Log. Usually you should use the `Major` (message transfer reports) level. But when you experience problems with the server kernel, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case low-level details will be recorded in the System Log as well. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.
> The kernel records in the System Log are marked with the `SYSTEM` tag.

> Kernel problems are very unlikely to happen. If you see any problem with the Server, try to detect which component is causing it, and change the Log setting of that component (Router, SMTP, POP, etc.) to get more information.

`Crash Recovery`
> If this option is enabled, the CommuniGate Pro Server uses special recovery techniques to proceed after various failures (including the crashing bugs in the Server software itself).

> If you see `"exception raised"` messages in your CommuniGate Pro Log and/or in the OS `system.log` or `mail.log`, you may want to disable this option and force the Server to stop when an exception is raised again, and to produce a *core dump* file. Core dump files can be uploaded to the CommuniGate Systems ftp site for examination.

> CommuniGate Systems recommends you to disable this option if you are running any beta-version of the CommuniGate Pro software.

`Separate WebAdmin Realms`
> If this option is disabled, WebAdmin realms are addressed using the `/Master/realmName/page` URLs, with `/Master/` as the authentication realm. The Server Administrator can access all WebAdmin pages by entering the password only once, but the Server Administrator Account must have the Master access right.

> If this option is enabled, WebAdmin realms are addressed using the `/realmName/page` URLs, with `/realmName/` as the authentication realm. The Server Administrator needs to enter a password to open each realm, but the Server Administrator needs the access right for that realm only.

> Enable this option if some of your Server Administrators do not have the Master access right.

Information fields

Information fields on the General Settings page display

- the name of the Server Operating System
- the Server hardware platform (CPU type)
- the CommuniGate Pro Server version
- the Server Local Time and Time Zone (this information is useful for Server Administrators that have to examine Logs from remote locations, as all time stamps in the System Logs are specified in the Server local time)
- the MAPI Connector server part version
- the flag indicating IPv6 support
- the Domain Name Server network address(es)
- the Server network addresses. Each address is accompanied with a link to the Domain it is assigned to, or the address is marked as un-assigned.
- the Server Startup Parameters.

Refresh

This button can be used after the Server OS local IP Addresses have been changed or the DNS settings for CommuniGate Pro Domains have been modified. When you click this button:

- the Server re-reads the list of Local IP Addresses from the OS
- the Server re-reads the Domain Name Server addresses from the OS settings
- the Server updates the "Assigned IP Addresses" for all Domains. If some Domains have IP Addresses specified "Using DNS A/MX Records", the new addresses are retrieved from the DNS system
- the Server re-loads the MAPI Connector server part (so you can upgrade the MAPI Connector server part without restarting the Server)

Drop Root

This button is available on certain Unix platforms. It allows the System Administrator to tell the server to drop the "superuser" privileges. Certain functions (such as OS Authentication, Execute Rules operations, etc.) may become unavailable.
If the Server succeeds to drop the "superuser" privileges, the button title changes to Restore Root. Click the Restore Root button to restore the "superuser" privileges.

---

# Specifying the Preferred Language

CommuniGate Pro supports multiple languages, and different users can use different languages. If most of your users will use the same language, it is recommended to set this language as the default one for the entire Server or for a particular Domain.

Use the WebAdmin Interface to open the Account Defaults page in the Users realm to specify Server-wide language settings. If you want to set a default language for a particular Domain, open that Domain pages in the WebAdmin Users realm, and open the Domain Account Defaults page from there.
Click the Preferences link to open the Default Preferences page.

Select the default Language and select a matching Preferred Character set: ISO-2022-JP for Japanese, KOI8-R for Russian, etc. If most of your users use modern Web browsers with the proper UTF-8 support, set the Use UTF-8 option to Reading and Composing.

Set the display names for the INBOX Mailbox and the virtual MAPI Outbox folder. These strings are used only with the CommuniGate Pro own client components - the WebUser Interface and MAPI, so you can enter any valid Mailbox name here, in any language. You can also change these names at any time.

Set the names for special Mailboxes - Sent, Drafts, Notes, Trash, Contacts, Calendar, and Tasks. Please note that these names will be used with the CommuniGate Pro own client components only - the WebUser Interface and MAPI. To make the user's IMAP clients use the same Mailboxes for the same purposes, the same Mailbox names should be specified in the IMAP client configurations. If you change these names later, the new Mailboxes will be created when a client needs to access a special Mailbox: the already existing special Mailboxes will not be renamed.

---

# Specifying the Preferred Time Zone

CommuniGate Pro supports multiple time zones, and different users can be located in different zones. If most of your users will use the same time zone, it is recommended to set this zone as the default one for the entire Server or for a particular Domain.

Open the Account Defaults page in the Domains section of the WebAdmin Interface if you want to set the Server-wide default time zone. If you want to set a default time zone for a particular Domain, open the Domains page of the WebAdmin Interface, open the Accounts or Settings page for the selected Domain and open the Domain Account Defaults page from there. Click the Preferences link to open the Default Preferences page.

Select the default Time Zone from the list. If you select the "built-in" zone (HostOS), the Server will use a fictitious zone that has the same time difference with GMT as the Server OS has at this time. This zone has no support for daylight saving time and it cannot be used for sending recurrent events outside your Server. Unless your Time Zone is not listed, avoid selecting the "built-in" zone.

# Base Directory Structure

All CommuniGate Pro Server files - Accounts, Domains, Mailboxes, settings, queues, etc. are stored in one place - in the Server *base directory*.

When the Server starts, it creates the following objects inside its base directory:

* The `Settings` directory. This directory contains files with module and kernel component settings.
* The `Queue` directory. This directory contains [Temporary and Message files](#). The Message files contain messages submitted to the Server, but still undelivered to all their recipients.
* `BadFiles` directory. This directory contains Message files the [Enqueuer kernel component](#) failed to parse. This directory should be empty.
* `Accounts` directory. This directory contains [account files](#) for the Main Server Domain.
* `Domains` directory. This directory contains directories for all other [Domains](#).
* `Submitted` directory. This drop-in directory is used to submit messages to Server via the [PIPE module](#).
* `SystemLogs` directory. This directory contains [Server Logs](#).
* `ProcessID` file. This file exists only when the Server is running and contains the numeric identifier of the Server process in the OS.
* `Directory` directory. This directory contains the Server [Central Directory](#) files.

For more information about the Account and Domain files and directories, see the [Objects](#) section.

You can use symbolic links to move some of these directories to other locations (and other disks).

# Command Line Options

The CommuniGate Pro Server supports the following command-line options (parameters):

`--CGateBase` *directory*
or
`--Base` *directory*
> The next parameter string specifies the location of the CommuniGate Pro *base directory*.

`--logToConsole`
> This option tells the Server to duplicate all its [System Log](#) records to the `stdout` (standard output). This option can be used for troubleshooting when the Web interface to System Logs is not available.

`--logAll`
> This option tells the Server to ignore all current Log Level settings and record all possible Log records.

`--daemon`
> This option can be specified on Unix platforms only. It tells the Server to fork and operate in the background, with stdin, stdout, and stderr redirected to `/dev/null`.

`--CGateApplication` *directory*
> The next parameter string specifies the location of the CommuniGate Pro *application directory*. You can use this option when the application itself cannot properly detect its own location, or if the CommuniGate Pro Server application file is not placed in the same location as other application directory files and subdirectories. For example, on OS/400 CommuniGate Pro Server is

located in an OS/400 library, and this parameter is used to tell the server where the Unix-style directory with WebUser, WebAdmin, WebGuide, and other files is located.

`--IPv6 [ YES | NO ]`
    See the [Network](#) section for the details.

`--lockLockFile [YES | NO]`
    This option tells the Server not to try to lock the ProcessID lock file. This option can be used if the file system hosting the *base directory* does not support file locks.

`--dropRoot`
    This option can be specified on Unix platforms only (this does not include Linux). It tells the Server to drop the *root privilege* permanently. The server drops the privilege approximately 60 seconds after the end of its kernel initialization process, so all listening sockets can be opened when the server is still running as the *root*. The root privilege cannot be restored later. See the [Server Root Privilege](#) section for more details.

`--delayOnStart` *seconds*
    This option tells the Server to sleep for the specified number of seconds before proceeding with initialization. This option can be used when CommuniGate Pro starts during system startup and it should let other services to startup fully (for example, an external file system can be mounted, a virtual IP address can be created, etc.)

`--threadsScope` *scope*
    This option can be specified on platforms supporting p-threads (OS/400 and most Unix flavors). The next parameter string can be either "system" or "process". See your OS manual to learn how these "scheduling scopes" work. If this option is not specified, the default OS scheduling mode is used.

`--batchLogon`
    This option can be specified on Microsoft Windows platforms only.
    The option tells the Server to use 'batch logon' instead of the 'network logon' when an account password is verified using the Windows OS password system.

`--sharedFiles`
    This option can be specified on Microsoft Windows and IBM OS/2 platforms only.
    The option tells the Server to open all files with the FILE_SHARE_READ sharing attribute making it possible for other programs (such as backup daemons) read the CommuniGate Pro *base directory* files when the server is running. This option is enabled by default on the Microsoft Windows NT/XP/200x platforms.

`--noSharedFiles`
    This option can be specified on Microsoft Windows and IBM OS/2 platforms only.
    The option tells the Server to open all files without the FILE_SHARE_READ sharing attribute if the Server does not need to read the file from several threads.

`--useNonBlockingSockets`
    This option tells the Server to set its TCP and UDP sockets in the non-blocking mode. This option can improve the Server performance on some platforms.

`--useBlockingSockets`
    This option tells the Server to set its TCP and UDP sockets in the blocking mode.

`--closeStuckSockets`
    This option tells the Server to maintain a list of open communication sockets and check if some socket operations did not complete in time and due to the kernel bugs the OS failed to interrupt the operation in time. It is recommended to use this option on heavily-loaded Solaris systems.

`--localIPBuffer` *size_value*
    This option tells the Server to use a buffer of the specified size when it retrieves a list of the Server Local IP addresses from the OS. On some platforms (such as Linux and Unixware) the default buffer size is set to a relatively small value, because some versions of these OSes have problems processing large buffers. If your Server system has many IP addresses (more than a thousand) and your CommuniGate Pro Server does not recognize all of them, you may want to use this parameter to specify a larger buffer size. The default size is 16K or 128K, you may want to specify larger values (`204800` or `200K`).

`--threadPriority [ YES | NO ]`
    If this option value is NO, the Server skips all attempts to increase an individual thread priority. Use this value if bugs in the Server OS cause an application to crash when a thread priority is increased ("non-global zones" in Solaris 10).

`--defaultStackSize` *size_value*
    This option modifies the default stack size (in bytes) for the process threads.

`--SIPUDPSendBuffer` *size_value*
`--SIPUDPReceiveBuffer` *size_value*
>   These options specify custom send and receive buffer sizes (in bytes) for the [SIP](#) UDP listener socket.

`--SIPUDPReceiverHighPty [ YES | NO ]`
>   If this option value is YES, the priority of the thread receiving SIP UDP packets is increased. It is recommended that you use this option only when a non-zero number of [SIP Enqueuer](#) threads is used.

`--DNRUDPSendBuffer` *size_value*
`--DNRUDPReceiveBuffer` *size_value*
>   These options specify custom send and receive buffer sizes (in bytes) for the [DNR](#) UDP socket.

`--DNRUDPReceiverHighPty [ YES | NO ]`
>   If this option value is YES, the priority of the thread receiving DNR UDP packets is increased.

`--excludeLocalIP` *ip_address*
>   Use these options to remove the specfied Network IP Address from the list of addresses which are processed as "local" (i.e. the addresses assigned to the Server computer).

`--createTempFilesDirectly` *pool_size*
>   This option modifies the way the Temporary Files Manager creates its files. With the default value of 0, a special thread is used to keep a pool of pre-created files ready for consumption by any component. If this option is set to a non-zero value, and the amount of pre-created Temporary Files in the pool is below this value, new Temporary Files are created with the requesting threads themselves.
>   You may want to specify a non-zero value for this option on heavily-loaded systems with low file creation performance (such as OpenVMS).

`--UseSystemPorts [ YES | NO ]`
>   On Unix systems, if this option value is YES, the Server will try to use the TCP/UDP ports with numbers below 1024, even when the Server application is not running as a ["superuser"](#) ("root").

`--randomDataDevice` *path*
>   This option specifies the path to the system entropy source. Without this parameter or when reading from the specified path fails the server uses the value of the system timer for seeding its PRNG.

Command line option names are case-insensitive.

## Specifying Command Line Options under Windows NT/200x/XP/Vista

You can specify the Command Line Options using the Services control panel "Startup Parameters" field. A non-empty set of Command Line Options is stored in the System Registry and it is used every time the CommuniGate Pro Messaging Server service is started without parameters. To clear the stored set of the Command Line Options, specify a single minus (`-`) symbol using the Services control panel "Startup Parameters" field.

## Customizing Unix Startup Scripts

You may need to add certain shell commands to the CommuniGate Startup script. Since the Startup script is a part of CommuniGate Pro application software, it is overwritten every time you upgrade your CommuniGate Pro system. Instead of modifying the Startup script itself, you can place a `Startup.sh` file into the CommuniGate Pro *base directory*. Startup scripts check if that file exists, and execute it before performing the requested start/stop operations.

## Customizing OpenVMS Startup Procedures

You may need to add certain DCL commands to the CommuniGate Startup procedure. Since the Startup procedure is a part of CommuniGate Pro application software, it is overwritten every time you upgrade your CommuniGate Pro system. Instead of modifying the Startup procedure itself, you can place a `STARTUP.COM` file into the CommuniGate Pro *base directory*. Startup procedure checks if that file exists, and it executes that file before starting the Server.

---

# Shutting Down

The CommuniGate Pro Server can be shut down by sending it a SIGTERM or a SIGINT signal.

On Unix and OpenVMS platforms you can use the startup script with the `stop` parameter, or you can get the Server process id from the ProcessID file in the base directory and use the `kill` command to stop the server. On OpenVMS platforms the KILL.EXE program can be found in the *application directory*.

On the Windows NT platform, you can use the Services control panel to stop and start the CommuniGate Pro server.

You can also use the `shutdown` CLI API command to stop the server.

When the Server receives a shutdown request, it closes all the connections, commits or rolls back Mailbox modifications, and performs other shutdown tasks. Usually these tasks take 5-15 seconds, but sometimes (depending on the OS network subsystem) they can take more time. Always allow the Server to shut down completely, and do not interrupt the shutdown process.

# OS syslog

The CommuniGate Pro server can store as much as several megabytes of Log data per minute (depending on the Log Level settings of its modules and components), and it can search and selectively retrieve records from the log. To provide the required speed and functionality, the Server maintains its own multithreaded Log system.

The Server places records into the OS log:

- when it starts up;
- when it shuts down;
- when it detects its own memory leaks;
- when it detects its own program error;
- when a program error exception (signal) is raised.

The system Log is:

- system.log or mail.log file on Unix systems
- Event Log on Windows systems

# Urgent Notifications

The CommuniGate Pro Server can display Urgent Notification Messages to Server Administrators.

The Urgent Notifications are displayed in WebAdmin Interface:

| MAIL.MYCOMPANY.COM | Settings | Users | Monitors | Directory | Master |
|---|---|---|---|---|---|
| *A problem with Module SMTP, TCP listener on [0.0.0.0]:25: failed to start. Error Code=network address/port is already in use* | | | | | |

There may be Notifications about failures with Helpers, filesystem errors, license keys expiration, and other critical events which require immediate reaction from the Server Administrator.
When there are several Notification in progress, one randomly choosen Notification is displayed.
The Server automatically deactivates the Notifications when they become outdated.

# Server Root Privilege

The CommuniGate Pro is designed as a highly secure application. In order to perform certain operations, the Server runs as *root* on Unix platforms, and it carefully checks that no user can access restricted OS resources via the Server. Since many other servers do not provide the same level of security, system administrators preferred to run servers in a non-root mode, so a hole in the server security would not allow an intruder to access the restricted OS resources.

CommuniGate Pro can "drop" the *root privilege*. The privilege can be dropped in the "permanent" or "reversible" mode. When asked to drop the root (uid=0) privilege, the Server changes its UID:

- to the UID of the Unix user `cgatepro` (if exists), otherwise

- to the UID of the Unix user `nobody` (if exists), otherwise
- to the UID 1

When the root privilege is dropped, the following restrictions apply:

- No Listener port with number < 1024 can be opened. If you try to add a listener with the port number n (n < 1024), the port with the number 8000+n is opened instead (unless the --UseSystemPorts Command Line Option is used).
- Remote applications started via Account-Level Rule EXECUTE command run in the current CommuniGate Pro Server environment (the effective UID and other OS-level process parameters are not changed).
- OS Passwords cannot be used.

If the root privilege was dropped in the "reversible" mode, the root privilege can be restored. For example, if you need to open a listener on the port 576, but the Server root privilege has been dropped, you should restore the root privilege first, then open the listener port, and then you can drop the Root privilege again.

To drop the root privilege permanently, use a special Command Line Option.

To drop the root privilege in the "reversible" mode, click the "Drop Root" button on the General page. The button should change to the "Restore Root" button - you can use it to restore the Server root privilege. This option is not available on those platforms that cannot drop the root privilege correctly (Linux).

# Domain Administrator

If your Server has several Domains, you may want to grant some users in those Domains the *Domain Administrator access right*.

A Domain Administrator can control the Domain using the same WebAdmin port (see HTTP module description for the details), or using the Command Line Interface (API) commands. Domain Administrator access is limited to his Domain (and, optionally, to certain other domains), and to explicitly allowed Domain and Account settings and operations.

When you grant the Domain Administrator access right to a user, you will see a list of specific access rights - the internal names of Domain and Account Settings.
Each option controls the settings this Domain Administrator can modify, and the operations this Domain Administrator can perform.

Domain Administrator access rights can be granted to users by a Server Administrator with the All Domains and Account Settings access right.

A System Administrator with the All Domains and Account Settings access right can perform all operations potentially available to a Domain Administrator in any Domain.

## Domains Administrators in other Domains

When a customer has several Domains, you may want to let an Account in one Domain administer other Domains. You should grant such an Account the `CanAdminSubDomains` access right. Then you should open the Domain Settings page for the target Domain and specify the Administrator's Domain name in the Administrator Domain Name field.

Sample:
A customer has the company1.com, company2.com, company3.dom Domains on your Server. You may want to specify `company1.com` as the Administrator Domain Name in the company2.com and company3.com Domain Settings. Now, any Account in the company1.com Domain that has the `CanAdminSubDomains` Domain Administrator right can administer all three Domains.

**Note:** when a Domain Administrator connects to the Domain WebAdmin Interface, the browser displays the Login Dialog Box. If the Administrator Account is in a different Domain, the full account name (*accountName@domainName*) should be specified.

## Domain Administrator Access Rights

Domain Administrators can perform operations on their own Domains and, optionally, on certain other Domains. The set of allowed operations is defined by the Domain Access Rights explicitly granted to the Domain Administrator Account and listed in the table below:

| Domain Settings | |
|---|---|
| **Access Right** | **Description** |
| `DomainAccessModes` | [Enabled Services](#) |
| `AutoSignup` | Provisioning: [Auto-Signup](#) Setting |
| `ExternalOnProvision` | Provisioning: [Consult External on Provision](#) Setting |
| `TrailerText` | Client Interfaces: [Mail Trailer Text](#) Setting |
| `WebBanner` | WebUser Interface: [Web Banner Text](#) Setting |
| `WebSitePrefix` | WebUser Interface: [Personal Web Site Prefix](#) Setting |
| `Foldering` | Large Domains: [Foldering Method](#) Setting |
| `FolderIndex` | Large Domains: [Generate Index](#) Setting |
| `RenameInPlace` | Large Domains: [Rename in Place](#) Setting |
| `AllWithForwarders` | Mail to All: [Send to Forwarders](#) Setting |
| `MailToAllAction` | Mail to All: [Distributed for](#) Setting |
| `ExternalOnUnknown` | Unknown Names: [Consult External for Unknown](#) Setting |
| `MailToUnknown` | Unknown Names: [Mail to Unknown Names](#) Setting |
| `MailRerouteAddress` | Unknown Names: [Mail Rerouted to](#) Setting |
| `SignalToUnknown` | Unknown Names: [Signal to Unknown Names](#) Setting |
| `SignalRerouteAddress` | Unknown Names: [Signal Rerouted to](#) Setting |
| `AccessToUnknown` | Unknown Names: [Access to Unknown Names](#) Setting |
| `AccessRerouteAddress` | Unknown Names: [Access Rerouted to](#) Setting |
| `CentralDirectory` | [Directory Integration](#) Setting |
| `CertificateType` | Security: [Domain PKI](#) Settings |
| `KerberosKeys` | Security: [Kerberos Keys](#) |
| `RelayAddress` | SMTP Sending: [Send via](#) Setting |
| `ForceSMTPAuth` | SMTP Receiving: [Force AUTH](#) Setting |
| `recipientStatus` | SMTP Receiving: [When Receiving](#) Setting |
| `ServiceClasses` | Can create, rename, and remove [Classes of Service](#) |
| Objects | |
| **Access Right** | **Description** |
| `CanCreateAccounts` | [Create](#), [rename](#), and [remove](#) Accounts |
| `CanCreateSpecialAccounts` | Create [single-mailbox](#) or [Legacy INBOX](#) Accounts |
| `CanCreateGroups` | Create, rename, remove, and modify [Groups](#) |
| `CanCreateForwarders` | Manage [Forwarders](#) |
| `CanCreateNamedTasks` | Manage [Named Tasks](#) |
| `CanCreateLists` | [Create](#), [rename](#), and [remove](#) Mailing Lists |
| `CanAccessLists` | Modify [Mailing Lists](#) |
| `CanCreateAliases` | |

| | Manage [Aliases](#) |
|---|---|
| `CanCreateTelnums` | Manage [Telephone Numbers](#) |
| `CanPostAlerts` | Post Domain and Account [Alerts](#) |
| `CanAdminSubDomains` | Administer [other Domains](#) |
| `CanModifySkins` | Manage [Domain Skins](#) |
| `CanModifyPBXApps` | Manage [Domain Real-Time Applications](#) |
| `CanAccessMailboxes` | Unrestricted Access to all [Account Mailboxes](#) |
| `CanAccessWebSites` | Unrestricted Access to all [File Storage](#) files |
| `CanControlCalls` | Unrestricted Access to all [Call Control](#) functions |
| `CanCreateWebUserSessions` | Manage WebUser sessions via [CLI](#) |
| `CanImpersonate` | Ability to [Impersonate](#) |
| `CanControlAirSync` | Ability to control [AirSync](#) clients |
| `CanCreditAccounts` | Ability to credit [Account Balances](#) |
| `CanChargeAccounts` | Ability to charge [Account Balances](#) and to reserve funds. |
| `CanChargeReserves` | Ability to charge fund reserved in [Account Balances](#) |
| **Account Settings** | |
| **Access Right** | **Description** |
| `ServiceClass` | [Class of Service](#) settings |
| `BasicSettings` | [Basic](#) Settings: Password, RealName, [Custom](#) and Public Info settings |
| `PSTNSettings` | [PSTN](#) settings |
| `WebUserSettings` | [Preferences](#) |
| `UseAppPassword` | CommuniGate Password: [Allow to Use](#) |
| `PWDAllowed` | CommuniGate Password: [Allow to Modify](#) |
| `PasswordEncryption` | CommuniGate Password: [Encryption](#) |
| `RequireAPOP` | Authentication methods: [Secure only](#) |
| `UseKerberosPassword` | [Kerberos](#) Authentication |
| `UseCertificateAuth` | [Certificate](#) Authentication |
| `UseSysPassword` | Authentication methods: [Enable OS Password](#) |
| `OSUserName` | Authentication methods: [Server OS user name](#) |
| `UseExtPassword` | Authentication methods: [External Authentication](#) |
| `LogLogin` | Logging for login/logout events in a [Supplementary Log](#) |
| `FailedLoginFlows` | Authentication: [Failed Login Limit](#) |
| `AccessModes` | [Enabled Services](#) |
| `MailInpFlow` | Mail Transfer options: [Incoming Mail Limit](#) |
| `MailOutFlow` | Mail Transfer options: [Outgoing Mail Limit](#) |
| `MaxMessageSize` | Mail Transfer options: [Incoming Message Size Limit](#) |
| `MaxMailOutSize` | Mail Transfer options: [Outgoing Message Size Limit](#) |

| | |
|---|---|
| MailToAll | Mail processing options: Accept Mail to all |
| AddMailTrailer | Mail processing options: Add Trailer to Sent Mail |
| QuotaNotice | Mail Quota Processing: Send Notice |
| QuotaAlert | Mail Quota Processing: Send Alerts |
| QuotaSuspend | Mail Quota Processing: Delay New Mail |
| RulesAllowed | Mail processing options: Rules |
| RPOPAllowed | Mail processing options: RPOP Accounts |
| MaxAccountSize | Mail Storage limits: Mail Storage |
| MaxMailboxes | Mail Storage limits: Mailboxes |
| DefaultMailboxType | Mail Storage options: New Mailboxes |
| MaxSignalContacts | Signal processing limits: Contacts |
| SignalRulesAllowed | Signal processing options: Rules |
| CallsLimit | Signals: Concurrent Calls option |
| CallLogs | Signals: Call Logs option |
| DialogInfo | Signals: Call Info option |
| CallInpFlow | Signals: Incoming Calls Limit option |
| CallOutFlow | Signals: Outgoing Calls Limit option |
| RSIPAllowed | Signals: RSIP Registrations |
| AirSyncAllowed | Ability to specify which AirSync clients can access the Account. |
| MaxRosterItems | Signals: MaxRosterItems option |
| IMLogs | Signals: IM Logs option |
| NotifyOutFlow | Signals: Outgoing NOTIFY Requests Limit option |
| MaxWebSize | File Storage limits: Web Storage |
| MaxFileSize | File Storage limits: Web Storage |
| MaxWebFiles | File Storage limits: Web Files |
| AddWebBanner | File Storage options: Add Web Banner |
| DefaultWebPage | File Storage options: Default Web Page |

# WebAdmin Preferences

Server and Domain administrators can customize the WebAdmin Interface parameters, including the initial number of objects to be displayed in the Object Lists, the refresh rate for the Monitor pages, etc. The Preferences also specify the character set used for WebAdmin pages. If you plan to use non-ASCII symbols, specify the correct character set first.

The bottom part of every WebAdmin page contains the name of the authenticated Administrator viewing that page, and the link to the WebAdmin Preferences page.

Each CommuniGate Pro WebAdmin *realm* has its own Preferences. Click the Preferences link to open the Preferences page.

The specified Preferences are stored as one of the Administrator Account Setting attributes, so different administrators can have different Preferences.

# Customizing Domain WebAdmin Interface

The Server Administrator can modify the look and feel of the Domain WebAdmin interface. For each CommuniGate Pro Domain, a custom version of WebAdmin files can be created.

The WebAdmin Interface uses the same [Skins](#) Interface as the WebUser Interface. The WebAdmin Interface uses the `Admin-xxxxx` Skins.
Within those Skins, the `adminyyyyyyy` files are used to compose pages in the User Realm of the Server WebAdmin Interface, as well as the Domain WebAdmin Interface pages.

To modify a the Domain WebAdmin Interface pages, upload custom `adminyyyyyyy` files into the `Admin-xxxxx` Skins. You can create new `Admin-xxxxx` Skins, and select those Skins (shown without the Admin- prefix) in the Domain Administrator Preferences.

The Server Administrator can also upload custom `admin*` files into the Server-wide and Cluster-wide Skins.

**Note:**The Server WebAdmin interface **always** uses the "stock" Skin files located in the WebSkins subdirectory of the *application directory*. If you modify the WebAdmin interface for the Main Domain, the modified pages will be used when a Domain Administrator of the Main Domain uses the WebAdmin Domain Interface.
The Server Administrator will see the Server WebAdmin Interface (with the Settings, Domains, Directory, and Monitors realms) and the "stock" Skin files will be used to compose the Server WebAdmin Interface pages.

# Customizing Server Prompts

The Server Administrator can modify the protocol prompts and other text strings the CommuniGate Pro Server sends to client applications.

To modify the Server Strings, open the General pages in the WebAdmin Interface Settings realm, and open the Strings page:

| Keyword | Text |
|---|---|
| ACAPByeBye | CommuniGate Pro ACAP closing connection |
| ACAPPrompt | CommuniGate Pro ACAP ^0 |
| FTPByeBye | CommuniGate Pro FTP Server connection closed |
| FTPPrompt | CommuniGate Pro FTP Server ^0 ready |
| | ................................ |
| SubjectFailed | Undeliverable mail |

**Note**: The actual Strings page has much more elements.

To modify a Server String, enter the new text in the text field, and select the lower radio button.
To change the string to its default value (displayed above the text field), simply select the upper radio button.

Click the Update button to update the Server Strings.

# CommuniGate Pro

| Sys Admin | Logs | Router | Protection | Security | PKI | Intercept | Scalability | Alerts | Statistics |

## Server Logs

- **Creating and Deleting Server Log Files**
- **Specifying a Time Interval**
- **Filtering Log Records**
- **Filtering by Prefix Key**
- **Searching**
- **Time Stamps and Time Zones**
- **Overflow Markers**
- **Preferences**
- **Sending to Remote syslog Servers**
- **Using a Trigger**
- **Supplementary Logs**
  - **Setting Updates**
  - **Call Detail Records (CDRs)**
  - **EMails**
  - **Login/Logout**
  - **Statistics**

All components of the CommuniGate Pro Server store messages in one unified Log.
Each record contains a time stamp, the log level, the tag identifying the component that created the record, and the record data itself.

CommuniGate Pro Logs are plain text files, and they can be processed with any text-processing utility.

When sending a support request to CommuniGate Systems technical support, always include a portion of the Log indicating the problem.

# Creating and Deleting Server Log Files

Use the WebAdmin Interface to examine the Server Logs. Open the Logs section in the Monitor realm. The Server page opens, it lists the stored Server Log files. The current Log is marked with the asterisk (∗) symbol.

You should have the Can Monitor Server Access Right to view the Server Logs.

The options on the top of the page allow you to specify when the Server Logs files are created and deleted:

| Log Manager Settings | | | |
|---|---|---|---|
| Start New File | Every: day | or if Larger than: | |
| Delete Old Files | In: 10 days | Time Precision: 3 | |
| External Logger | Records to Send: All Info | Server Address: | |
| | Facility Code: mail | Source IP Address: [192.168.1.152]  : | |

```
Start New File
```
A new file is created automatically every day (at midnight), or more often, according to this setting value.

```
if Larger than
```
A new Log file is also created if the current Log file size exceeds the specified limit.

The Log files are created in the `SystemLogs` subdirectory of the Server *base directory*.

`Delete Old Files`
> Shortly after a new Log file is created, the Server checks all files in the `SystemLogs` subdirectory, and removes all files that are older than the time period specified with this setting.

`Time Precision`
> This setting specifies how many digits should be used in log record time stamps to specify fractions of a second.

`External Logger`
> Please see the [Sending to Remote Servers](#) section.

You should have the `CanTuneLoggerSettings` [Monitor Access Right](#) to modify the Logs Engine settings.

You can select one or several Logs in the list and then remove them using the Delete Marked Logs button. The active (current) Log file cannot be deleted.

You should have the `CanDeleteLogs` [Monitor Access Right](#) to delete Logs.

If there are too many Log files on the Server, you can enter a string in the Filter field and click the Display button: only the Logs with names matching the Filter string will be displayed:

| Filter: | 11 of 11 selected |
|---|---|
| **Name** | **Size** |
| voicemail_prompt | 355K |
| * 2015-12-07_14-39 | 5974K |
| 2015-12-07 | 31M |
| 2015-12-06_23-20 | 5707K |
| 2015-12-06_13-19 | 31M |
| 2015-12-06_02-05 | 31M |
| 2015-12-06_00-54 | 31M |
| 2015-12-06_00-52 | 31M |
| 2015-12-06_00-49 | 31M |
| 2015-12-06_00-48 | 31M |
| 2015-12-06_00-46 | 31M |

Click the Log file name to open it.

# Specifying a Time Interval

When the Log appears in your browser window, all Log records are displayed. Since Logs can have thousands of records, you may want to view only a portion of the Log. Interrupt the Log downloading process and specify the Log Level and the Time Interval options:

| | | | |
|---|---|---|---|
| Log Level: | All Info | | **Full Screen** |
| Filter: | | Keyed | RegEx |
| Server Time: 4:41:53PM | Interval: | - | |

Only the records with time stamps in the specified interval are displayed.

If you are viewing the current Log and specify "*" in the second field, all records placed in the Log by this moment are displayed.

If you are viewing the current Log and specify some future time in the second field, the Server will keep the browser channel open, sending new Log records as they are placed in the Log. The channel is closed either at the end of the specified Time Interval, or when the Server starts a new Log.

# Filtering Log Records

The CommuniGate Pro Logs can be very big, reaching several hundred megabytes of data on a heavily loaded Server or on a Server with low-level logging enabled.
It is difficult to examine an entire Log of that size.

`Level`
> Use this setting to suppress displaying records that are more detailed than the specified value (have a higher level marker).

`Filter`
> Use this option to specify the Filter string. Only the records containing this string will be displayed.
> The first part of log records (including a time stamp and a level marker) is not used for filtering operations.

`RegEx`
> If this option is selected, the Filter string is interpreted as a *regular expression*.

Click the Display button to display only the records that contain the specified substring.

Example:

> Some of your users complain that sometimes their mailer application cannot retrieve messages from your server properly and that they see error messages informing them about some protocol errors.

> Since it does not occur often, you should run the IMAP module with its Log Level set to All-Info, though this will make your Logs very big. Finally, a user contacts you and says that the mailer has just displayed the same error.

> You open the Log and set the Level to 3 (Problems). Now you may see all the problems with the IMAP module that occurred today. When you find the record that indicates the problem your user is talking about, you see that that record has the `IMAP-437425` tag. So, you type IMAP-437425 into the Filter field, and change the Log Level to 5 (All Info). As a result, you see a clean log of that particular IMAP session.

| Log Level: | All Info | | **Full Screen** |
|---|---|---|---|
| | Filter: | Keyed   RegEx | |
| Server Time: 4:41:53PM | Interval: | - | |

```
00:06:23.261 4 IMAP-437425([64.173.55.175]) got connection on
[64.173.55.169:143](mail.communigate.com) fr
00:06:23.261 5 IMAP-437425([64.173.55.175]) out: * OK CommuniGate Pro IMAP
Server 5.1.8 at mail.commun
00:06:23.261 5 IMAP-437425([64.173.55.175]) inp: 1 CAPABILITY
00:06:23.261 5 IMAP-437425([64.173.55.175]) out: * CAPABILITY IMAP4 IMAP4REV1
ACL NAMESPACE UIDPLUS ID
00:06:23.266 5 IMAP-437425([64.173.55.175]) inp: 2 AUTHENTICATE METHOD
AAAAAAAAAAAAAAAAAAAAA=
00:06:23.268 2 IMAP-437425([64.173.55.175]) 'user@domain.com' connected from
[64.173.55.175:31358]
00:06:23.268 5 IMAP-437425([64.173.55.175]) out: 2 OK completed\r\n
00:06:23.269 5 IMAP-437425([64.173.55.175]) inp: 3 LIST "" "*"
00:06:23.269 5 IMAP-437425([64.173.55.175]) out: * LIST (\UnMarked) "/"
Calendar\r\n* LIST (\Marked) "
00:06:23.279 5 IMAP-437425([64.173.55.175]) inp: 4 SELECT "Tasks"
00:06:23.270 5 IMAP-437425([64.173.55.175]) out: * FLAGS (\Answered \Flagged
\Deleted \Seen \Draft $MD
00:06:23.272 5 IMAP-437425([64.173.55.175]) inp: 5 UID SEARCH NOT DELETED
00:06:23.272 5 IMAP-437425([64.173.55.175]) out: * SEARCH 32 49 76 84 94 96 98
100 101 102 113 116 117
00:06:23.275 5 IMAP-437425([64.173.55.175]) inp: 6 UID FETCH 193 (BODYSTRUCTURE
FLAGS)
00:06:23.275 5 IMAP-437425([64.173.55.175]) out: * 35 FETCH (BODYSTRUCTURE
(("text" "calendar" ("chars
00:06:23.278 5 IMAP-437425([64.173.55.175]) inp: 7 UID FETCH 193
(BODY.PEEK[HEADER])
00:06:23.278 5 IMAP-437425([64.173.55.175]) out: * 35 FETCH (BODY[HEADER]
```

```
{722}\r\ncontent-class: urn:
00:06:23.280 5 IMAP-437425([64.173.55.175]) inp: 8 UID FETCH 193 (BODY.PEEK[1])
00:06:23.280 5 IMAP-437425([64.173.55.175]) out: * 35 FETCH (BODY[1]
{539}\r\nBEGIN:VCALENDAR\r\nMETHO
00:06:23.281 5 IMAP-437425([64.173.55.175]) inp: 9 UID FETCH 191 (BODYSTRUCTURE
FLAGS)
00:06:23.281 5 IMAP-437425([64.173.55.175]) out: * 34 FETCH (BODYSTRUCTURE
(("text" "calendar" ("chars
```

# Filtering by Prefix Key

The Keyed option instructs the Server to scan the Log twice. First, it scans the Log (within the specified Time Interval) and finds all records matching the filter string. These strings are not displayed, but their Prefix Keys are remembered. The Prefix Key is the first part of the record (not including the time stamp and the level marker) till the first space symbol. Up to 100 different Prefix Keys are remembered.

Then the Log is scanned again (within the specified Time Interval), and the Server displays all records that have Prefix Keys matching one of the remembered Prefix Keys.

Some protocols (such as SIP) do not use connections. A SIP session ("dialog") consists of several packets (each packet is recorded with its own SIPDATA-*NNNNNN* Prefix Key), but all packets contain the same Call-ID line. Use the

: Call-ID:*caller-id*

filter string with the Keyed option to display all SIP session packets:

| Log Level: | All Info | | | **Full Screen** |
| --- | --- | --- | --- | --- |
| | Filter: | | Keyed   RegEx | |
| Server Time: 4:41:53PM | Interval: | - | | |

```
00:54:10.312 2 SIPDATA-000502 out: req udp [10.0.0.1]:5060 REGISTER(680 bytes)
sip:node6.communigate.com
00:54:10.312 5 SIPDATA-000502 out: REGISTER sip:node6.communigate.com SIP/2.0
00:54:10.312 5 SIPDATA-000502 out: Via: SIP/2.0/UDP
64.173.55.170:5060;branch=z9hG4bK234
00:54:10.312 5 SIPDATA-000502 out: Max-Forwards: 69
00:54:10.312 5 SIPDATA-000502 out: From: <sip:usrname@node6.communigate.com>
00:54:10.312 5 SIPDATA-000502 out: Call-ID: 72D532E1CEB813B537E4E44058354C68-
2494453@node9.communigate.com
00:54:10.312 5 SIPDATA-000502 out: Contact:
<sip:299@node9.communigate.com;services=no>;expires=90
00:54:10.312 5 SIPDATA-000502 out: CSeq: 114249520 REGISTER
00:54:10.312 5 SIPDATA-000502 out: User-Agent: CommuniGatePro-gateway/5.1.4
00:54:10.312 5 SIPDATA-000502 out: Authorization: Digest
realm="ns.communigate.com",username="usrname",non
00:54:10.312 5 SIPDATA-000502 out: Expires: 90
00:54:10.312 5 SIPDATA-000502 out: Content-Length: 0
00:54:10.312 5 SIPDATA-000502 out:
00:54:10.328 2 SIPDATA-000503 inp: rsp udp [64.173.55.167]:5060 200-REGISTER(566
bytes)
00:54:10.328 5 SIPDATA-000503 inp: SIP/2.0 200 OK
00:54:10.328 5 SIPDATA-000503 inp: Via: SIP/2.0/UDP
64.173.55.170:5060;branch=z9hG4bK234
00:54:10.328 5 SIPDATA-000503 inp: From:
<sip:usrname@node6.communigate.com>;tag=9B5A8DB531C3FD7A
00:54:10.328 5 SIPDATA-000503 inp: To:
<sip:usrname@node6.communigate.com>;tag=7FBB267A3903E5B0
00:54:10.328 5 SIPDATA-000503 inp: Call-ID: 72D532E1CEB813B537E4E44058354C68-
2494453@node9.communigate.com
00:54:10.328 5 SIPDATA-000503 inp: CSeq: 114249520 REGISTER
00:54:10.328 5 SIPDATA-000503 inp: Expires: 90
00:54:10.328 5 SIPDATA-000503 inp: Contact:
<sip:299@node9.communigate.com;services=no>;expires=90
00:54:10.328 5 SIPDATA-000503 inp: Event: registration
```

```
00:54:10.328 5 SIPDATA-000503 inp: Date: Mon, 16 Mar 2015 08:53:04 GMT
00:54:10.328 5 SIPDATA-000503 inp: Allow: PUBLISH,SUBSCRIBE
00:54:10.328 5 SIPDATA-000503 inp: Allow-Events: presence,message-
summary,reg,keep-alive
00:54:10.328 5 SIPDATA-000503 inp: Supported: path
00:54:10.328 5 SIPDATA-000503 inp: Server: CommuniGatePro/5.1.4
00:54:10.328 5 SIPDATA-000503 inp: Content-Length: 0
00:54:10.328 5 SIPDATA-000503 inp:
00:54:10.328 2 SIPDATA-000503 sent to SIPC-000234
00:55:25.328 2 SIPDATA-000507 out: req udp [10.0.0.1]:5060 REGISTER(680 bytes)
sip:node6.communigate.com
00:55:25.328 5 SIPDATA-000507 out: REGISTER sip:node6.communigate.com SIP/2.0
00:55:25.328 5 SIPDATA-000507 out: Via: SIP/2.0/UDP
64.173.55.170:5060;branch=z9hG4bK236
00:55:25.328 5 SIPDATA-000507 out: Max-Forwards: 69
00:55:25.328 5 SIPDATA-000507 out: From:
<sip:usrname@node6.communigate.com>;tag=35270A39FB68F573
00:55:25.328 5 SIPDATA-000507 out: To: <sip:usrname@node6.communigate.com>
00:55:25.328 5 SIPDATA-000507 out: Call-ID: 72D532E1CEB813B537E4E44058354C68-
2494453@node9.communigate.com
00:55:25.328 5 SIPDATA-000507 out: Contact:
<sip:299@node9.communigate.com;services=no>;expires=90
00:55:25.328 5 SIPDATA-000507 out: CSeq: 114249521 REGISTER
00:55:25.328 5 SIPDATA-000507 out: User-Agent: CommuniGatePro-gateway/5.1.4
00:55:25.328 5 SIPDATA-000507 out: Authorization: Digest
realm="ns.communigate.com",username="usrname",non
00:55:25.328 5 SIPDATA-000507 out: Expires: 90
00:55:25.328 5 SIPDATA-000507 out: Content-Length: 0
00:55:25.328 5 SIPDATA-000507 out:
00:55:25.343 2 SIPDATA-000508 inp: rsp udp [64.173.55.167]:5060 200-REGISTER(566
bytes)
00:55:25.343 5 SIPDATA-000508 inp: SIP/2.0 200 OK
00:55:25.343 5 SIPDATA-000508 inp: Via: SIP/2.0/UDP
64.173.55.170:5060;branch=z9hG4bK236
00:55:25.343 5 SIPDATA-000508 inp: From:
<sip:usrname@node6.communigate.com>;tag=35270A39FB68F573
00:55:25.343 5 SIPDATA-000508 inp: To:
<sip:usrname@node6.communigate.com>;tag=7EF99B799DFD7632
00:55:25.343 5 SIPDATA-000508 inp: Call-ID: 72D532E1CEB813B537E4E44058354C68-
2494453@node9.communigate.com
00:55:25.343 5 SIPDATA-000508 inp: CSeq: 114249521 REGISTER
00:55:25.343 5 SIPDATA-000508 inp: Expires: 90
00:55:25.343 5 SIPDATA-000508 inp: Contact:
<sip:299@node9.communigate.com;services=no>;expires=90
00:55:25.343 5 SIPDATA-000508 inp: Event: registration
00:55:25.343 5 SIPDATA-000508 inp: Date: Mon, 16 Mar 2015 08:54:19 GMT
00:55:25.343 5 SIPDATA-000508 inp: Allow: PUBLISH,SUBSCRIBE
00:55:25.343 5 SIPDATA-000508 inp: Allow-Events: presence,message-
summary,reg,keep-alive
00:55:25.343 5 SIPDATA-000508 inp: Supported: path
00:55:25.343 5 SIPDATA-000508 inp: Server: CommuniGatePro/5.1.4
00:55:25.343 5 SIPDATA-000508 inp: Content-Length: 0
00:55:25.343 5 SIPDATA-000508 inp:
00:55:25.343 2 SIPDATA-000508 sent to SIPC-000236
```

# Searching

Use your browser Find command to search for a string in the filtered portion of the CommuniGate Pro Log.

Use the Print command of your Web browser to print the filtered Log.

# Time Stamps and Time Zones

Each Log record has a time stamp indicating when the record was created. The time is displayed using the local time ("GMT shift") of the CommuniGate Pro Server used when the Log file was created.

If the Server OS uses the time zone with daylight saving time, the time stamps used in the Log will not change when the local time ("GMT shift") changes. The new local time will be used when the new Log file is created.

# Overflow Markers

The CommuniGate Pro Log Manager is designed as high-speed engine capable of processing thousands records per second, without delaying the execution of the Server component that generated the Log records. When some component generates a huge amount of records, (most likely, due to the Log Level set for that component), even the Log Manager may be unable to store all those records in the Log file.

If a new record cannot be placed into the Log due to a Log Manager performance problem, the Log Manager stores a short Overflow Marker instead. The Overflow Marker is a line with three asterisk (***) symbols.

If you filter the Log, the displayed part of the Log will always contain the OverFlow Markers if they exist in the selected part of the Log. If several sequential Overflow Markers have to be displayed, only the first one is displayed.

# Preferences

Administrators can specify their individual Log Viewer Preferences.

Use the Preferences link to open the Monitor Preferences.

| Log Manager | | |
|---|---|---|
| Log Panel Height:  500 | Open showing last:  default(2 min) | Display Limit:  default(3M) |

`Open showing last`
> When you open the currently active Log file, this setting specifies the inital starting time of the Time Interval (see below), so you see only the recent Log records.
> When you open an inactive Log file, the Time Interval is not initialized, and the Log is displayed from the beginning.

# Sending to Remote `syslog` Servers

You may want to send CommuniGate Pro Log records to an external `syslog` server.
Usually these servers are not providing the CommuniGate Pro Log Manager performance, so you should send only a small part of the Log records to those servers.

Use the following settings to configure remote logging:

`Records to send`
> Specify the level of Log Records to be sent to a remote syslog server. Records that are more detailed than the specified value (have a higher level marker) are not sent.

`Server address`
> Specify the IP address of the remote syslog server. If you do not specify the port number, the standard port number 514 is used.

`Facility Code`
> The remote syslog server can store log records in different locations based on this code value.

`Source IP Address`
> Specify the local IP address and/or the port to send messages to the syslog server. If an address or a port is not specified explicitly, the server OS will assign them.

If the Log Manager fails to open a UDP socket or fails to send a datagram to the selected remote syslog server, the Log Manager suspends sending records to the remote syslog server till the end of the current second.

# Using a Trigger

You may want to use a [Trigger Handler](#) to send notifications when a Crash-level record is added to the CommuniGate Pro Log.

Open the [Statistics Elements](#) page in the WebAdmin Interface, and specify a Trigger Handler for the `logLastCrashRecord` element. When a Crash-Level (0-Level) record is added to the Log, the selected Trigger Handler is invoked.

**Note:** the Trigger Handler is started once in 5-10 seconds. If more than one Crash-Level records were written to the CommuniGate Pro Log during this time period, the notification message contains only the last Crash-Level record.

---

# Supplementary Logs

The CommuniGate Pro Server maintains supplementary Logs in addition to the main Server Log described above.

Supplementary Logs are designed to store the most important events of a certain type only.

Supplementary Log records are stored as text file lines: one record is stored on one line. Each record/line starts with a timestamp prefix:
`hh:mm:ss.ddd`
where *hh* is the hour, *mm* is the minute, *ss* is the second, and *ddd* is the millisecond of the moment when the record was generated.

New supplementary Log Files are created daily.

Use the WebAdmin Interface to examine supplementary Logs. Open the Logs section in the Monitor realm. Click to select the supplementary Logs type. A page opens, it lists the stored supplementary Log files. The current Log is marked with the asterisk (`*`) symbol.

You should have the [Can Monitor](#) Server Access Right to view the supplementary Logs.

The supplementary Log files are created inside subdirectories of the `SystemLogs` subdirectory.

## Setting Updates

These Log files contain a record for each Server module or component setting update.

The following record format is used:
*component userName newSettings*
Where:

`component`
     the updated component name
*userName*
     the name of the Administrator Account used to update the Settings.
*newSettings*
     the updated settings [object](#) value (usually - a dictionary).

## Call Detail Records (CDRs)

These Log files contain Call Detail Records in various formats. See the [Real-Time Signals](#) section for more details.

## EMails

The E-mail Transfer [Dequeuer](#) component generates a record for every E-mail delivery event. The following record format is used (the tabulation symbol is used as the field separator):
`01 state returnPath fromAddress authAddress submitAddress localIP queueID messageID messageSize origRecipient module(queueName)recipient report`
Where:

`01`
     record format version

*state*

the message delivery result: RELAY (relayed), FINAL (delivered to the final destination), ERROR (a fatal error, resending will not help), TMPER (a temporary error condition, can be resent later).

*returnPath*

message envelope return-path

*fromAddress*

the message From: address

*authAddress*

if the message sender was authenticated, the authenticated Account name. Otherwise, an empty string

*submitAddress*

the name of the component submitted the message, followed by the Network Address associated with the message source.

*localIP*

the Server Local Network Address used to submit the message

*queueID*

the message internal Queue UID (numeric)

*messageID*

the message Message-Id header field value

*messageSize*

the message size in bytes

*origRecipient*

the delivery target, as originally specified in the submitted message

*module*(*queueName*)*recipient*

the actual delivery target: the deliver module name, the name of the Module queue (remote domain for the SMTP Module, Account name for the Local Delivery Module, etc.), optionally followed with the specific address (remote recipient for the SMTP Module, supplementary address for the Local Deliver module, etc.)

*report*

the report delivery string. As it can be a multi-line one, it is stored as the text presentation of the report [String](#) object

## Login/Logout

Records are placed into these Log files when users log into their Accounts, and when they log out.

The following record formats are used (the tabulation symbol is used as the field separator):
```
01 login accountName [address]:port service method
01 logout accountName [address]:port service
01 login verify [address]:port service method
```
Where:

01

record format version

login, logout, verify

operation. For session-based protocols (such as [POP](#) or [XMPP](#)), the login and logout records are created. For session-less protocols (such as [SIP](#)), the verify records are created.

*address*, *port*

the IP address the user client has connected from

*service*

the name of the [Service](#) (protocol) associated with this operation.

*method*

the name of the [Authentication Method](#) used (for the login and verify operations only).

## Statistics

These Log files contain values of all Statistics Elements. See the [Statistics](#) section for more details.

Sys Admin | Logs | **Router** | Protection | Security | PKI | Intercept | Scalability | Alerts | Statistics

# Router

- **Address Structure**
- **Main Domain Name**
- **Domains and DNS Records**
- **Routing Table**
  - **Prefix**
  - **Sample**
  - **Route**
  - **Domain-Level Records**
  - **Account-Level Records**
  - **All-Local Records**
- **Special Addresses**
- **Explicit Routing via Remote Systems**
- **Routing to Real-Time Applications**
- **Phone Number Routing**
- **Routing by IP Addresses**
- **Routing via Modules**
- **External Helper Routing**
- **ENUM Routing**
- **`tel:` Routing**
- **Default Records**
- **Extending Non-Qualified Domain Names**
- **All-Domain Aliases**
- **Cluster-wide Routing Table**

This section is for advanced administrators only. In most situations the default routing methods are sufficient. Only if your Server is working as a message or signal relay for other systems, or if you want to use sophisticated routing schemes, should you read this section.

When the Server processes a submitted Message, it extracts the information about recipients from the message "envelope" and decides into which module queue the message should be placed, which entity the module should send the message to, and how it should address that entity. Similar operations are performed with Real-time Signals received from external sources or generated with internal components and directed to local or external entities.

Access modules (such as POP, IMAP, WebUser Interface, etc.) also deal with addresses. When a client application or program logs in, it provides a name of the Account it wants to log into. This address is processed using the same operations as ones used to process Message and Signal addresses.

The CommuniGate Pro Router component implements these routing operations. Whenever your Server gets some address, that address is processed using the Router component. This provides additional consistency to all Server components: when, for example, you create an Alias for some Account, that Alias can be used to send E-mail and Signals to that Account, and the Alias name can be used to log into that Account.

# Address Structure

Each E-mail or Signal address consists of two strings: a domain name and a local part. Usually, an address looks like xxxx@yyyyy, where yyyyy is the domain name (the unique name of the recipient mail system) and xxxx is the local part, i.e. a user name or an account name in that system.

Addresses can be more complicated, for example, an E-mail address can include some path information:

```
<@zzzz:xxxx@yyyyy> or zzzz!yyyyy!xxxx or xxxx%yyyyy@zzzz
```
These addresses specify that an E-mail message or a Signal should be sent to the system zzzz first, and then that system should deliver it to xxxx@yyyyy (to the account xxxx on the system yyyyy).

When the Router parses an address, it extracts the name of the system the message should be delivered to. It becomes the domain name part of the address. The rest of the address is placed into the local part, i.e. the local part defines the recipient when the message or the signal is delivered to the system specified with the domain name. In the examples above, the domain name part is zzzz, while the local part is xxxx@yyyyy.

See the RFC822 and related documents for details on E-mail address formats.

If a local part contains a complex address (i.e. the local part itself contains domain name(s) and a local part), the local part is presented using the '%' notation: local%domain1%domain2, so the full address in the CommuniGate Pro canonical form would be `local%domain1%domain2@domain`.

# Main Domain Name

When the domain name is extracted from an address, the Router compares it against the domain name of the Server (set in the General settings). If they match, the domain name is set to an empty string. When the domain part becomes an empty string, the Router restarts processing with the local part, trying to divide it into the domain and local parts again.
For example, if the Main Domain name of your Server is company.com, then the following addresses will be converted as shown:

| Address | local part | domain part |
|---|---|---|
| support@company.com | support | company.com |
| -- *routed* --> support | | |
| | | |
| <@company.com:sales@example.com> | sales@example.com | company.com |
| -- *routed* --> sales@example.com | | |
| -- *routed* --> sales | example.com | |

# Domains and DNS records

Your Server can support many independent Domains in addition to the Main Server Domain.

In order to process Messages and Signals directed to your Server Domains, you should ensure that the Messages sent to that domain are directed to your Server with the global DNS system.

Example 1: your server (example.com) serves the example.com Domain and a partners-example.com Domain. Make sure that DNS MX records are created for both Domains, and that those records point to your example.com Server.

Example 2: your server (example.com) acts as an "Remote POP" mail host relay for some client systems. Each client has its own domain name (client1.com, client2.com, and client3.com), and you have configured your Router to ensure that all E-mails sent to the client1.com domain will be routed to the Unified Domain-Wide Account client1, etc.
You should also ensure that when an E-mail is sent to the client1.com domain, that E-mail is routed to your server (example.com). The client1.com MX record should be created in the DNS system and that record should point to your Server (example.com).

# Routing Table

When an address is parsed and its domain part is extracted, the Router checks the routing records in the Routing Table.

Use the WebAdmin Interface to manage the Routing Table. Open the Router page in the Settings realm:

Log Level:    Failures

Each line in the Routing Table is a routing record. A routing record contains optional prefixes, a left part, the equals (`=`) symbol and a right part. The semicolon (`;`) symbol can be used to place a comment after the right part of a routing record. A comment line can be added to the Table by inserting a line starting with the semicolon symbol.

The Router takes a parsed address (i.e. the domain and local parts of the address) and uses the Table, scanning its records from top to bottom. If an applicable record is found, it is applied as described below and the modified address is processed with the Router again.

`Log Level`
> Use this setting to specify what kind of information the CommuniGate Pro Router should put in the System Log. Usually you should use the `Failures` (address routing failures) level. But when you experience problems with the Router, you may want to set the Log setting to `Low-Level` or `All Info`: in this case more low-level information about the Router activity will be recorded in the System Log as well. The Router component records in the System Log are marked with the `ROUTER` tag.

## Prefix

A Routing record can contain a Relay-mode prefix: `Relay:` (can be shorten to `R:`), `NoRelay:` (can be shorten to `N:`), or `RelayAll:`. See the [Protection](#) section for the details.
If none of these prefixes is specified, the `NoRelay:` prefix is assumed.

A Routing record can contain zero, one, or several of the following operation-type prefixes:

- `Mail:` (can be shorten to `M:`). Records with this prefix take effect when an address is routed for some E-mail delivery operation.
- `Access:` (can be shorten to `A:`). Records with this prefix take effect when an address is routed for some account access operation.
- `Signal:` (can be shorten to `S:`). Records with this prefix take effect when an address is routed for some Signal operation.

These prefixes should be specified after an optional Relay-mode prefix. If none of these prefixes is specified, the Routing record is applied to addresses used in all operations.

## Sample

The left part of a Routing record contains a Sample: a string with an optional wildcard.

The following wildcards are supported:

`*`
> this wildcard matches zero or more symbols.

> Example:
>> `sta*r`
>> This sample matches any `sta`*XXXXXX*`r` string where *XXXXXX* is any substring (including an empty substring inside the `star` string).

`(`*size type*`)`
> *type* is the substring type:

- `d` - decimal digits (`0 .. 9`)
- `h` - hexadecimal digits (`0 .. 9, A .. F, a .. f`)
- `L` - alpha-numeric symbols (`0 .. 9, A .. Z, a .. z`)
- `*` - any symbols

*size* is an optional substring length. It can be specified in the following forms:

- *nnn* (where *nnn* a decimal number): a matched substring should have *nnn* symbols.
- *nnn+* : a matched substring should have *nnn* or more symbols.
- *nnn-mmm* (where *mmm* a decimal number, *mmm* >= *nnn*): a matched substring should have between *nnn* and *mmm* symbols.

Example:
    `sta(3*)r`
This sample matches any `sta`*XXX*`r` string where *XXX* are any 3 symbols.

Example:
    `sta(4+d)r`
This sample matches any `sta`*DDDDD*`r` string where *DDDDD* are 4 or more decimal digits.

Example:
    `sta(3-5h)r`
This sample matches any `sta`*HHH*`r` string where *HHH* are 3,4, or 5 hexadecimal digits.

The backslash (\) symbol is used as an escape symbol: \\ is processed as a single backslash symbol, \* is processed as an asterisk symbol, etc.

Only one wildcard symbol is allowed in a sample.

## Route

The right part of a Routing record contains a Route: a string with an optional `*` wildcard.

When the Record Sample matches an address, the address is changed to the Record Route. The substring matching the Sample wildcard substitutes the Route wildcard.

The backslash (\) symbol is used as an escape symbol: \\ is processed as a single backslash symbol, \* is processed as an asterisk symbol, etc.

Only one wildcard symbol is allowed in a Route.

## Domain-Level Routing Records

If the left part of a Routing record contains a domain name, the record specifies domain-level routing.

When some address is being processed and the domain name matches a domain name specified in such a record, the domain part is substituted with the right part of the routing record.

Example:
    `hq.company.com = twisted.company.com`
All addresses with the hq.company.com domain part are changed to have the twisted.company.com domain part. The Router restarts, trying to route the modified address.

A routing path can specify relays.

Example:
    `hq.company.com = hq.company.com@relay.company.com`
All E-mails and Signals directed to the domain name hq.company.com will be redirected to the system (domain)

relay.company.com, and then, from that system, to the domain hq.company.com.

If E-mails and Signals to several domains should be routed in the same or similar way, you may use the asterisk (`*`) symbol as the wildcard symbol.

Example:
    `*.old_company.com = new_company.com`
    In this case addresses in all the domains ending with .old_company.com will be changed to have the new_company.com domain part, with the local parts (user names) unchanged.

Very often this type of routing is used to process all subdomains of the some domain.

Example:
    `*.mycompany.com = mycompany.com`
    If the mycompany.com is the Server's Main Domain name, then this routing record makes the Server process E-mails and Signals sent to all subdomains of its Main Domain as E-mail and Signals sent to the Main domain, the address `user@mail.mycompany.com` will be processed as the `user@mycompany.com` address.

Example:
    `*.old_company.com = *.new_company.com`
    When such a routing line is entered and an E-mail or a Signal comes in for the domain `host5.old_company.com`, it is routed to `host5.new_company.com`.

Example:
    `system-*.mycompany.com = uu*.local`
    This routing line will redirect `system-abc.mycompany.com` to `uuabc.local` .

Besides domain-level routing records, routing for domains can be specified using account-level records (see below). Records for Unified Domain-Wide Accounts are domain-level routing records, too.

## Account-Level Routing Records

If the left part of a routing record contains an address in the angle brackets (`<` and `>`), the record is an Account-level record - a routing rule for a specific address.

When an address is parsed and the Router scans the Table records, it compares the address domain part with the domain part of all Account-Level routing records.

If the domain parts match, the Router compares the local part of the address with the local part of the account-level record address. If both domain and local parts match, the right part of the account-level routing record is used as the new address. The Router restarts, parsing and processing this new address.

**Note:** Because the Server Main Domain Name in the parsed address is immediately replaced with an empty string, account-level records that should apply to addresses in the Main Domain should not contain any domain part at all.

In the all examples below `mycompany.com` is the Server Main Domain name.

Example:
    `<sales> = bill`
    in this case, all messages to `sales@mycompany.com` will go to `bill`, as if they were sent to `bill@mycompany.com`.

**Note:** if there is an account-level record for the local name `xxxx`, there is no need to actually create a real `xxxx` Account on the Server. Additionally, that Account would be useless, since no message or signal will ever reach that Account: everything directed to the `xxxx` name will be routed elsewhere.

The right side of an account-level record can be any address.

Example:
    `<sales> = Bill@thatcompany.com`

All messages directed to sales@mycompany.com will be directed to Bill@thatcompany.com. The Router takes the new address, extracts the domain name (thatcompany.com) and local (Bill) parts, then the Router restarts trying to find a route to thatcompany.com.

You can use the wildcard symbol (*) in the local part of account-level records. The same symbol can be used in any part of the right-side address to specify substring substitution.

Example:
```
<dept-*> = postmaster@*-dept.mycompany.com
```
This record will redirect all E-mails and Signals sent to dept-sales@mycompany.com to the user postmaster at the sales-dept.mycompany.com department server.

You can use Router account-level records to reroute E-mails and Signals sent to some of the your Server Secondary Domains. In the following example, the client.com is a local Secondary Domain.

Example:
```
<sales@client.com> = Bill@client.com
```
All messages directed to sales@client.com will be directed to Bill@client.com.

Example:
```
<sales@client.com> = Bill
```
All messages directed to sales@client.com will be directed to Bill@mycompany.com (i.e. to the address Bill in the Main Domain).

In most cases you do not have to use account-level Router Records: if you need to provide an alternative name some Account, use Account Aliases instead. If you need to re-route all E-mails and Signals sent to some name in a local Server Domain to some other address, use Forwarders instead.

You may need to create an alias for a specific account on a foreign system. For example, if all E-mail sent to some domain should be routed to a specific mail host or to a unified account, but certain accounts in that domain should be routed to accounts on your or other systems.

Example:
```
Mail:<sales@client1.com> = sales-client1
Mail:client1.com = new.client1.com
```
These records route all E-mails directed to the account sales at the domain client1.com to the Account sales-client1 in your Server Main Domain, while messages to all other accounts in the client1.com domain are routed to the new.client1.com system.

The wildcard symbol (*) can be used only in the local part of the full account name (i.e. it can be used before the @ symbol).

You can use the wildcard feature to host several domains in one CommuniGate Pro Domain creating a unique "address space" for each domain name.

Example:
```
<*@client5.com> = cl5-*
<*@client7.com> = cl7-*
```
E-mails and Signals sent to sales@client5.com address will be directed to the cl5-sales Account in the Main Domain, E-mails and Signals sent to info@client5.com address will be directed to the cl5-info Account in the Main Domain, while E-mails and Signals sent to sales@client7.com address will be directed to the in the cl7-sales Account in the Main Domain.

This method can be used when you do not want to create full-scale CommuniGate Pro Domains for many domains containing very few Accounts.

## All-Local Routing Records

Account-level records can use wildcard symbol (*) as the domain part. This records are applied to the addresses that contain any local Domain (i.e. a Domain created on this CommuniGate Pro Server or Cluster). The right-side address of such record specifies an address

in the same local Domain.

Example:
    `<abuse@*> = postmaster`
    This record reroutes an abuse@*domainX.dom* address into postmaster@domainX.dom for all *domainX.dom* Domains created in
    your CommuniGate Pro system.

If the right-side address contains a domain part, the address is routed to that domain.

Example:
    `<abuse@*> = postmaster@somedomain.com`
    This record reroutes abuse@domainX.dom addresses into the postmaster%somedomain.com@domainX.dom addresses for all
    domainX.dom Domains created in your CommuniGate Pro system. Then the postmaster%somedomain.com@domainX.dom
    addresses are rerouted to the postmaster@somedomain.com address.

The local part of the left-side address can contain a wildcard (as in regular account-level records). The string matching this wildcard
symbol can be used in the right-side address.

Example:
    `<+*@*> = 011*`
    This record reroutes the `+490088899@domainX.dom` address into the `011490088899@domainX.dom` address for all domainX.dom
    Domains created in your CommuniGate Pro system.

---

# Special Addresses

If an address is routed to one of the local Domains (including an empty domain part which is routed to the Main Domain), the local
part is checked against the following special values:

`null`
    this address is directed to a fictitious internal "Black hole" module.
    When an E-mail message is routed to the "Black hole", the address is marked as "delivered" immediately, without any additional
    processing.
    When a Signal request is routed to the "Black hole", request is considered "delivered" immediately, generating the 200 "OK"
    response, unless it is an `INVITE`, `UPDATE`, or `OPTIONS` request: in these cases, the 480 "No Address Found" response is generated.

    Example:
        `<*@bad.company.com> = null`
        `<junk> = null`
        With these records in the Routing Table, the Server will discard all E-mails and Signals sent to the domain
        `bad.company.com`, as well as all E-mails and Signals sent to the Main Domain address junk.

`MAILER-DAEMON`, `Calendar-Alarm`
    these special addresses are processed in the same way as the `null` address.
`error`
    E-Mails and Signals sent to this address are rejected without processing, generating an error report/response. If the domain name
    part of an address is `error`, or if the domain name part is empty, and the local name part is `error`, the address is rejected without
    processing, generating an error report/response.

    Example:
        `bad.company.com = error`
        `<junk> = error`
        With these records entered, the Server will reject all E-mail messages and Signals sent to the bad.company.com domain or
        to the Main Domain address junk.

`blacklisted`
    E-mails sent to this address are rejected without processing, generating the "Blacklisted Address" error report.
    See the [SMTP module](#) description for the details.
`spamtrap`

E-mails and Signals sent to this address are rejected without processing, but the [SMTP module](#) processes them in a special manner. See the [Protection](#) section for the details.

`incomplete`

Signals sent to this address are rejected with the "Address Incomplete" error code. It can be used to support some SIP clients that send call (INVITE) requests as soon as the user dials a digit.

Example:
```
<(1-4d)@*> = incomplete
```
With these Router record present, dialing a 1-,2-, or 3-digit number will return the "Address Incomplete" error code to the client, forcing it to wait for additional input from the user.

`listserver`

E-mails sent to this address are sent to the [LIST](#) module.

`all`

E-mails sent to this address are sent to all Domain Accounts. See the [Domains](#) section for the details.

`alldomains`

E-mails sent to this address are sent to all Accounts in all Domains. See the [Domains](#) section for the details.

If an address is routed to the domain name `null`, then it is processed in the same way as the addresses routed to the local part `null` in a local domain (see above).

If an address is routed to the domain name `error`, then it is processed in the same way as the addresses routed to the local part `error` in a local domain (see above).

If an address is routed to the domain name `blacklisted`, then it is processed in the same way as the addresses routed to the local part `blacklisted` in a local domain (see above).

If the domain name part ends with the symbols `.here` or `._here`, this suffix is removed, and the remaining part of the domain name is used as the name of a local CommuniGate Pro Domain. This suffix allows you to avoid routing loops in certain situations.

Example:
```
dept1.xyz.com = dept1.xyz.com._here
dept2.xyz.com = dept2.xyz.com._here
*.xyz.com = *.abc.com
```
E-Mails and Signals sent to all subdomains of the xyz.com domain are rerouted to the subdomains of the abc.com domain, except for the addresses in the dept1.xyz.com and dept2.xyz.com subdomains, which are routed to the local dept1.xyz.com and dept2.xyz.com CommuniGate Pro Domains.

---

# Explicit Routing via Remote Systems

After all Routing Table records are applied, the Router checks if the domain name part ends with the `._via` suffix. If the suffix is found, it is removed, the shorten domain name is used as the target host name, and the local part of the address is used as the address to pass to that host.

Example:
The Server Main Domain name is `company.com`.
E-Mails and Signals sent to the `sales.company.com` Domain should be relayed to a separate sales.company.com server, while addresses in all other subdomains of the company.com domain should be processed as addresses in the Main Domain, i.e. user@subdomain.company.com addresses should be processed as user@company.com addresses.
You can implement these routing rules using the following records:

```
sales.company.com = sales.company.com._via ; explicitly relay to a remote host
*.company.com     = company.com            ; all other subdomains are rerouted
```

**Note:** addresses in the sales.company.com domain will be relayed with the domain part removed, i.e. the address <user@sales.company.com> will be relayed to sales.company.com host as <user>.
This may cause troubles if the sales.company.com server does not accept addresses without domains. See the next example for a possible solution.

Example:

E-Mails and Signals sent to the client1.com, client2.com, and client3.com domains should be sent to the host.com server. You can implement this routing using the following records:

```
client1.com = client1.com@host.com._via
client2.com = client2.com@host.com._via
client3.com = client3.com@host.com._via
```

or, in a more flexible way:

```
client1.com = client1.com@relay

client2.com = client2.com@relay

client3.com = client3.com@relay

relay       = host.com._via
```

Note: You can specify just `host.com` instead of `host.com._via` here (given there is no other router record for host.com), but in this case mail to `user@client1.com` will be sent to the host.com as `user%client1.com@host.com`. By specifying the `._via` suffix you not only tell the Router to route the address to a relaying module, but you also force that module to send only the local part of the address to the remote server.

### Address Processing without the `._via` suffix

| | | |
|---|---|---|
| `user @ client1.host` | Router converts to | `user%client1.host @ relay` |
| `user%client1.host @ relay` | Router converts to | `user%client1.host @ host.com` |
| `user%client1.host @ host.com` | Router stops | *no rule for host.com* |
| `user%client1.host @ host.com` | Router accepts | for SIP/SMTP `host.com` host as `user%client1.host@host.com` |

### Address Processing with the `._via` suffix

| | | |
|---|---|---|
| `user @ client1.host` | Router converts to | `user%client1.host @ relay` |
| `user%client1.host @ relay` | Router converts to | `user%client1.host @ host.com._via` |
| `user%client1.host @ host.com._via` | Router accepts | for `host.com` as `user@client1.host` |

If the domain part of the address contains the `._via` suffix, the module checks the last domain name part after removing this suffix. If that part is a number, the dot (`.`) symbol separating this part is changed to the colon (`:`) symbol:

`host.domain.dom.26._via` --> `host.domain.dom:26` When the domain name contains a colon symbol, the SIP, XMPP, and SMTP modules:

- Do not use DNS MX/SRV records for that name, and retrieve the DNS A-records only.
- Use the number after the colon symbol as the number of the TCP/UDP port to connect to, instead of using the standard port number (25 for SMTP, 5060 for SIP).

The Router also checks if the domain part of the address ends with the `._relay` suffix. This suffix is removed and the resulting domain name is used as the target host name (after changing the optional port number separator to the colon symbol).
This domain name (after removal of the optional port number and its separator) is added to the local name, using the `@` symbol as the separator.

Example:
    The Server Main Domain is `company.com`.
    E-mails and Signals for the `xxxx.department.company.com` domains (where xxxx can be sales, marketing, etc.) should be sent to separate servers, according to their DNS records, while addresses in all other subdomains of company.com should be processed as addresses in the Main Domain, i.e. user@subdomain.company.com addresses should be processed as user@company.com addresses.
    You can implement this routing using the following records:

```
*.sales.company.com = *.sales.company.com._relay ; explicitly relay outside
*.company.com       = company.com                ; all other subdomains are rerouted
```

The Router also checks if the domain part of the address ends with the `._dir` suffix. This suffix is processed the same way as the `_via` suffix. In addition the server tries to retrieve the record from the [Directory](#) via the `dn` built from the original address and the destination domain in the form `mail=user@domain, cn=host.com`.

If the record can be retrieved (this may require setting up a [Remote Directory Unit](#) for the `cn=host.com` search base) the result of address routing is accepted, otherwise the routing error is generated.

If the attempt to retrieve the record results in some connection-level error, the routing result is a temporary error, so, for example, SMTP senders would retry deliveries to this address.

# Routing to Real-Time Applications

Many Signals (especially phone calls) should be handled with the "stock" or custom [Real-Time Applications](#). To Route a Signal to an Application you need to specify the Application name and, separated with the hash (`#`) symbol, the name of the Account that will be used to run the application:

The following record routes Signals sent to the *someName@someDomain* address to the application `myProgram` started on behalf of the `user@domain` Account:

    <someName@someDomain> = myProgram#user@domain

You can specify the application parameters by adding them after the application name, enclosed into the `{` and `}` brackets and separated with the comma (`,`) symbol.

The following record routes Signals sent to the *someName@someDomain* address to the application `myProgram` started on behalf of the `user@domain` Account. The program is started with 2 parameters - "mixer" and "fast":

    <someName@someDomain> = myProgram{mixer,fast}#user@domain

The wildcards in the right part of a Router record are substituted before any processing begins, so you can use the wildcard value as the application parameter.

The following record routes Signals sent to the `+(20d)@someDomain` addresses to the application `myProgram` started on behalf of the `user@domain` Account. The program is started with a parameter. The parameter is the catenation of the string `num=` and the wildcard value - the phone number without the leading `+` symbol:

    <+(20d)@someDomain> = myProgram{num=*}#user@domain

# Phone Number Routing

If the domain part of an address is `telnum`, the address local part is processed as a E.164 phone number.

The following steps are taken by the Router **before** it applies its Routing Table(s) and other routing methods:

- The Router checks if the phone number is assigned to any Account in any CommuniGate Pro Domain. If such an account is found, the Signal is redirected to that Account. See the [Accounts](#) section for more details on Assigned Phone Number.
- The Router applies the [ENUM Routing](#) method, using all Telephony ENUM DNS domains specified. In a [Dynamic Cluster](#) environment, the Server-wide ENUM domains are checked first, then the Cluster-wide ENUM domains are used.

If the phone number is not rerouted by any of the above methods, the Router processes it as a regular address.

| Phone Number ENUM Domains |
|---|
| |

To add a ENUM domain, enter its name into the empty field, and click the Update button.
To remove a ENUM domain, remove its name from the field and click the Update button.
Domains are used in the specified order.

See the [PSTN](#) section to learn more about PSTN and telephone number routing.

---

# Routing by IP Addresses

After all Routing Table records are applied, the Router checks if the domain name is actually an IP address. If the IP-address domain name is not enclosed into the square brackets, the Router encloses it: `user@10.34.45.67` is converted into `user@[10.34.45.67]`. This allows you to specify Routing Table records for IP addresses assuming that the address is always enclosed into square brackets.

For IP addresses enclosed in square brackets, the Router checks if the IP address is assigned to one of this Server Domains. If a Domain is found, the IP address is substituted with that Domain name. If the IP address is the IP address of the Server Main Domain, an empty string is placed into the domain name part, and the Router makes the next iteration after parsing the local name part of the address.

If an IP address is not assigned to a local Domain, the Router processes the `[10.34.45.67]` domain name as the `10.34.45.67.default_port._via` name:
the Router sends the address to the SIP or SMTP module, cutting off the domain part and using it as the host name to relay to.

---

# Routing via Modules

If no Routing Table record can be applied to an address, and the address is not a special address or an IP address of a local domain, the Router calls each communication module requesting a routing operation.

Each module looks at the address passed and can:

- ignore the address if the module does not know how to handle it;
- modify the address (for example, the LIST module converts addresses `listname-admin@listdomain` into the real address of the mailing list owner);
- reject the address (for example, the Local Delivery module rejects `username@domainname` addresses if the domain name is a name of a local domain, and there is no username Account or Alias in that domain);
- accept the address.

If a module has modified an address, the Router makes a new iteration, repeating all steps for the new, modified address.

If the Router is called from the [Message Enqueuer](#) component, and a module has accepted the address, the message is enqueued to this module for delivery.

If the Router is called from the [Signal](#) component, and a module has accepted the address, the Signal Request is sent to this module for processing.

Each module is called twice. First, the Router calls each module asking to process "explicit" addresses. On this call the modules process only the addresses that are definitely directed to that module: the SMTP module processes addresses with the domain part ending with `._smtp`, the LIST module processes the addresses of the existing mailing lists, etc.

If all modules have ignored an address, the Router calls each module again, asking for a "final" attempt. On that stage, the Local Delivery module processes all addresses directed to local Domains, the SIP module accepts all signal-type addresses, the SMTP module processes all addresses with domain names that have at least one dot, etc.

This two-step method allows several modules to correctly process addresses without relying to a particular module call order. If each module would process an address in one step, `listname@domainname` addresses (that look like Local account addresses), would be rejected with the Local Delivery module if it is called before the LIST module, `user@accountName.local` addresses would be taken with the SMTP module instead of the Local Delivery module, etc.

See the module descriptions for the details.

---

# External Helper Routing

After all Routing Table records are applied, the Router checks if the domain name is the `external` string. In this case, the domain part

is cut off, and the local part is passed to the [External Authenticator](#) program.

The external program can use any method to process the supplied address, and it should return a modified address or an error code.

If a modified address is returned, the Router makes the next iteration with this new address.

Example:
Signals to addresses starting with `011` in any local Domain should be routed using an external Helper program.
You can implement this routing using the following record:
```
NoRelay:Signal:<011*@*> = tele-*@external ; route using an external program
```

If a Signal is sent to `0115556666@local.domain.dom`, where local.domain.dom is a local Domain, the address will be rerouted to `tele-5556666@external` and the External Helper will receive a request to route the `tele-5556666` address.

---

# ENUM Routing

The Router supports DNS-based routing for telephone numbers. This method is usually applied to *E.164 numbers* - telephone numbers starting with the plus symbol followed with the country code, area code, and the local number.

If the domain name has the `._enum` suffix, then the Router:

- follows RFC2916 and issues the DNS NAPTR request for the number in the address local part, using the domain name (without the `._enum` suffix) as the search suffix.
- processes all result records of the `E2U+SIP` type for Signal-type addresses, or the E2U+EMAIL for other addresses.
- if the found mapping string starts with the `sip:`, `sips:`, `mailto:` prefix, removes the prefix.

The Router restarts, processing the found mapping string as the new destination address.

If the DNS search returns the "unknown host name" error, the `._enum` domain name suffix is replaced with the `._noenum` suffix, and the Router restarts to process the modified address.

Example:
Router records:
```
<+(d)@*> = +*@telnum ; direct +number to e164 "telnum" domain
telnum = e164.arpa._enum ; direct +number to e164.arpa
e164.arpa._noenum = pstn
<+44*@pstn> = gatewaycaller{+44*}#pbx
pstn = main.office.dom
```

A sample address `+14153837164@some.local.domain.dom` will be processed using the following steps:

- The first Router record converts this address to `+14153837164@telnum`
- The second Router record converts this address to `+14153837164@e164.arpa._enum`
- The Router looks for a DNS NAPTR record `4.6.1.7.3.8.3.5.1.4.1.e164.arpa`
- The resulting record `sip:pbx@communigate.com` is found, the `sip:` prefix is removed, and the Router restarts to process the `pbx@communigate.com` address

A sample address `+14155551212@some.local.domain.dom` will be processed using the following steps:

- The first Router record converts this address to `+14155551212@telnum`
- The second Router record converts this address to `+14155551212@e164.arpa._enum`
- The Router looks for a DNS NAPTR record `2.1.2.1.5.5.5.5.1.4.1.e164.arpa`
- There is no NAPTR record for this domain name, so the address is converted to `+14155551212@e164.arpa._noenum` and the Router restarts to process this address
- The third Router record converts the address into `+14155551212@pstn` and the Router restarts to process this address
- The fourth Router record directs all calls to the `+44` country code to the local `gatewaycaller` PBX application, but this record does not match the `+14155551212@pstn` address
- The fifth Router record redirects this address to `+14155551212@main.office.dom` so the Signal is relayed to the `main.office.dom` server.

# `tel:` Routing

When a `tel:`*phoneNumber* URI is being parsed, it is converted into `sip:`*phoneNumber*`@tel` URI internally. The fictitious `tel` domain is usually routed to the `telnum` domain (see above) to provide unified handling of all phone numbers.

When composing a URI from an internal form, any `sip:`*phoneNumber*`@tel` URI is composed as `tel:`*phoneNumber*.

# Default Records

When the CommuniGate Pro Server is installed, the following records are placed into the Routing Table:

`<root> = postmaster`
> This record reroutes all E-mails and Signals sent to the `root` name to the postmaster Account.
> This is useful on Unix systems, where many logging utilities are preconfigured to mail reports to the user `root`.

`localhost =`
> On many systems the `localhost` domain name used for the local IP address of that system, and some mailer programs use this name as a domain name.
> This record routes addresses within the "localhost" domain to the main server domain.

`mailhost =`
> Some mailer programs use the `mailhost` name as the domain name of the local mail server.
> This record routes such addresses to the Accounts in the CommuniGate Pro Main Domain.

`<blacklist-admin*@blacklisted> = postmaster`
> This record implements White Hole processing for blacklisted hosts.

`<\*(3-4d)@*> = voicemail#*`
> This record redirects all Signals (calls) sent to *\*nnn* and *\*nnnn* addresses in any local Domain to the PBX Voicemail application started for the Account with the *nnn* alias.

`<7(2d)@*> = pbx{*}#pbx`
> This record redirects all Signals (calls) sent to *7nn* addresses in any local Domain to the pbx Account in the same Domain.
> This Router record is needed to implement certain functions of the stock PBX Center application.

`<8(3d)@*> = pickup{*}#pbx`
> This record redirects all Signals (calls) sent to *8nnn* addresses in any local Domain to the pbx Account in the same Domain, starting the `pickup` application. The application then routes the *nnn@callerDomain* address and "picks up" an incoming call pending for that Account.
> See the PBX Services section for the details.

`tel = telnum`
> This record is used to process `tel:`*phoneNumber* URIs (see above).

`<+(8-20d)@*> = +*@telnum`
> This record redirects all *+nnnn...nn* addresses in any local Domain to the fictitious `telnum` domain.

`Signal:telnum = pstn`
> This record redirects all Signals (calls) sent to the fictitious domain `telnum` to the fictitious `pstn` domain.

`Signal:<*@pstn> = gatewaycaller{*}#pbx`
> This record redirects all Signals (calls) sent to the fictitious `pstn` domain to the `gatewaycaller` application started on behalf of the `pbx` Account in the Main Domain.
> The phone number (the local part of the address in the `pstn` domain) is passed to the application as a parameter.

`Signal:<911@*> = emergency@localhost`
> This record redirects all Signals (calls) sent to the `911` addresses in any local Domain to the `emergency` name in the `localhost` domain (this name is usually Routed to the Main Domain).

`Signal:<112@*> = emergency@localhost`
> This record redirects all Signals (calls) sent to the `112` addresses in any local Domain to the `emergency` name in the `localhost` domain (this name is usually Routed to the Main Domain).

```
Signal:<emergency> = emergency#pbx
```
> This record redirects all Signals (calls) sent to the `emergency` addresses in the Main Domain to the `emergency` application started on behalf of the `pbx` Account in the Main Domain.

```
Signal:<(7d)@*> = localAreaCall{*}#pbx@localhost
```
> This record redirects all Signals (calls) sent to 7-digit numbers in each local Domain to the `localAreaCall` application started on behalf of the `pbx` Account in the Main Domain.
> The phone number (the local part of the address) is passed to the application as a parameter.

All these default records can be modified or removed, if needed.

# Extending Non-Qualified Domain Names

Users working on sites that have many different Servers (`server1.myorg.org`, `server2.myorg.org`, `server3.myorg.org`) tend to use addresses with non-qualified domain names (`user@server1`, `user@server2`, `user@server3`). When you have only few servers in your myorg.org "upper level" domain, you can "fix" those addresses by specifying several Router Table records:

```
server1 = server1.myorg.org
server2 = server2.myorg.org
server3 = server3.myorg.org
```

If you have many servers in your myorg.org "upper level" domain, it becomes impossible to provide Router Table records for all of them. In this case you may want to enable the Add `myorg.org` to Non-Qualified Domain Names option. If this option is enabled, and an address cannot be routed using CommuniGate Pro Router Table and Modules, and the domain part of the address does not contain a dot symbol, the specified string (`myorg.org`) is added to the address domain name (separated with the dot symbol). The address `user@someserver` will be converted to the `user@someserver.myorg.org` address and the Router will try to route this new, corrected address.

**Note:** It is a very bad practice to use non-qualified domain names in E-mail or Signal addresses. Enable this option only if you can not enforce a policy that requires your users to specify correct, fully-qualified domain names in all addresses they use.

# All-Domain Aliases

| All-Domain Aliases | |
| --- | --- |
| **Local Name** | **Reroute to** |
| | |

This table allows you to specify aliases that will work for all local Domains.

When the CommuniGate Pro Server detects that a message or a signal should be directed to some name in one of the Server local domains, these records are checked. If the local part of the address matches the Local Address field in one of these records, the address is rerouted to the address specified in the Reroute To field.

If, for example, the `abuse` and `postmaster@maindomain.dom` addresses are entered into the All-Domain Aliases table (as shown above), then all messages directed to any `abuse@domain.dom` address (where *domain.com* is one of the CommuniGate Pro Domains) are rerouted to the `postmaster@maindomain.com`.

**Note:** it is very easy to create routing loops using these records: if you enter

```
postmaster -> postmaster@maindomain.dom
```

into this table, you will create a loop that will make it ***impossible to connect to the Server as `postmaster`***. If you want E-mails and Signals to all `postmaster` names in all Domains to go to the postmaster account in the main CommuniGate Pro domain, you should use:

```
postmaster -> anyname@postmaster.local
```

or, if the [Direct Mailbox Addressing](#) option is enabled:

```
postmaster -> mailboxName#postmaster
```

You can use wildcard (`*`) symbols in these fields.

For example, you may want to create a "dial plan" for your organization that has 10 different departments, each served with its own Domain:

| All-Domain Aliases | |
|---|---|
| **Local Name** | **Reroute to** |
| | |
| | |
| | |
| | |

If Accounts in each Domain have aliases in the `200-299` range, then the users can call other users within the same Domain by dialing the `2`*xx* number.
They dial the `91` prefix (a `912`*xx* number) to reach users in the domain1.com Domain.
They will use the `92` prefix to reach users in the domain2.com Domain, etc.

# Cluster-wide Routing Table

The CommuniGate Pro [Dynamic Cluster](#) maintains the Cluster-Wide Routing Table. When you open the Router WebAdmin page on any Cluster member, you see the link that opens a Cluster-Wide Routing Table page. All modifications made to this Table are automatically propagated to all Cluster Members.

The Cluster-Wide Router Table is processed as an extension of the Server Router Table: the Cluster-Wide Router Table records are checked when no Server Router Table record can be applied.

# Protection

- **Prohibiting Unauthorized Relaying**
  - **Specifying Client IP Addresses**
  - **Specifying Client Domains by Name**
  - **Configuring the SMTP module**
- **Client-only Logins**
- **Relaying for Mobile (non-client) Users**
  - **The SMTP AUTH method**
  - **The Read-then-Send method**
  - **Account and Domain Settings**
- **Return-Path Address Verification**
- **Blacklisting Offenders**
  - **Specifying Offender Addresses**
  - **Using DNS-based Blacklisting (RBL)**
  - **Blacklisting Domains by Name**
  - **Un-listing Addresses (White Hole Addresses)**
  - **Processing Messages from Blacklisted Addresses**
  - **Temporarily Blocked Addresses**
- **Checking Network Address Status**
- **Spam Traps**
- **Banning Mail by Header and Body Lines**
- **Filtering Mail**
- **Relaying Rerouted Messages**
- **Cluster Setup**

The Internet is flooded with soliciting E-mail messages distributed to millions of E-mail addresses. These messages are known as "spam".

Spammers fill your user Mailboxes with a huge amount of unwanted messages, not only overloading your network and Server resources, but making mail retrieval very slow and difficult for your users.

Besides the methods described in this section, additional methods are described in the Denied Addresses section.

# Prohibiting Unauthorized Relaying

If your SMTP module can accept incoming TCP connections, your Server can be used by spammers as a mail relay engine: they can distribute their messages all over the world using your Server as an *open relay*.

Also, if your SIP module can accept incoming SIP requests, your Server can be used by "voip" spammers as a SIP relay engine: they can distribute their calls and/or instant messages all over the world using your Server as an *open relay*.

To protect your site from spammers, you should restrict the Server relaying functionality. Basically, only your own users should be

able to use your Server to relay E-mail messages and Signal requests to other places on the Internet. Messages and Signal requests coming from other sources should go only to your own Accounts, and should be relayed to other Internet sites only when you have explicitly allowed that type of relaying.

## Specifying Client IP Addresses

The simplest way to decide if an incoming SMTP message or a SIP request is coming from your own user is to look at the network (IP) address it is coming from. If all your users connect from one or several LAN(s), you can treat all messages coming from those networks as "messages from Clients", and your Server will relay them to the Internet.

Use the WebAdmin Interface to open the Network pages inside the Settings section (realm), and click the Client IP Addresses link.

Enter the IP addresses on your client connect from, as well as the IP addresses of other systems that should be allowed to use your server as a mail relay:

| Client IP Addresses |
|---|
| |
| Process LAN IP Addresses as Clients |

Process LAN IP Addresses as Clients
    Select this option to include all LAN IP Addresses into the Client IP Addresses list.

The IP addresses are specified in a multi-line format. See the Network section for more details.

If you provide dial-up services, enter the IP address ranges you have allocated to your dial-up users.

## Specifying Client Domains by Name

You can specify your Client IP Addresses using the *reverse lookup* domain names.

| Detect Clients by DNS Name |
|---|
| |

**Note:** each Domain can have its own Client IP Addresses list, extending the Server-wide and Cluster-wide lists.

When a client connects from an IP address not listed in the Client IP Addresses list, and the Detect Clients by DNS Name option is enabled, the server tries to get the domain name for that IP address (if the IP address is $aa.bb.cc.dd$, the Server tries to retrieve the PTR record for the $dd.cc.dd.aa.in-addr.arpa$ name). If the PTR domain name is retrieved, it is checked against the strings specified in the table (these strings can include the wildcard (*) symbols). If the retrieved name matches one of the table strings, the server retrieves the DNS A record for the retrieved domain name, and checks that the IP address is included into the IP addresses in that record. If it is included, the address is considered to be a "Client IP Address", and it is processed in the same way as if it was entered into the Client IP Addresses list.

**Note:** while this method was popular with legacy mail servers, it can be very expensive for large-scale systems. It requires your Server to make 2 DNS transactions for each incoming connection not coming from explicitly specified Client IP Addresses, and these transactions can take a lot of time.
Use this method only when absolutely necessary, for example when your Server needs to support a large (and unknown) set of campus networks, and the only thing known about those networks is the fact that all their IP addresses can be "reversed-resolved" into some subdomain of the school domain. Even in this case, try to enter all known addresses and networks into the Client IP Addresses list,

decreasing the number of required "reverse-resolving" operations.

## Configuring the SMTP module

When a message is received with the SMTP module, and the sender IP address is not found in the Client Addresses list, the message is marked as being received "from a stranger". If this message should be relayed by your server to some other host on the Internet, and that host is not listed in the Client IP Addresses list either, the message can be rejected.

As a result, servers and workstations included into the Client Addresses list can use your Server to send (relay) messages to any mail server on the Internet. But any message coming from an unlisted address and directed to some other unlisted system can be rejected. This will prohibit spammers from using your Server as an "open mail relay".

Since this functionality can affect your legitimate users if you do not specify their IP addresses correctly, the Relay to non-Clients option is available on the SMTP Relaying page.
Set that option to "if received from `Clients`", and "stranger-to-stranger" relay attempts will be rejected.

The Client IP Addresses list can include addresses of some other mail servers. The Server can relay mail sent by anybody and addressed to a server with a network address included into the Client IP Addresses list, but it can also check if the message address is a "simple" one.

The SMTP Relaying page contains the To Client IP Addresses option. Set this option to "If Sent to: `simple` addresses" to prohibit relaying of "complex addresses" (such as `username%somehost@otherserver`) to servers listed in the Client IP Addresses list. This setting will prevent spammers from using your servers for "two-server relays".

The following is the "two-server relay" method:

- a spammer sends a message with the `username%somehost@server2` address to the `server1` server;
- the `server1` server relays the message to the `server2` server, because the `server2` address is included into the `server1` Client IP Addresses list;
- the `server2` server relays the message to username@somehost, since it has received it from `server1`, which is included into the `server2` Client IP Addresses list.

When servers relay only "simple" E-mail addresses to each other, those servers cannot be used for "two-server relaying" even if they maintain "mutual trust" (i.e. list each other in the Client IP Addresses lists).

To avoid problems with old mail servers that ignore the quote marks in addresses, the addresses with the local part containing quotes cannot be relayed to Client IP Addresses servers if the "simple" option is selected.

If the Relay to Client IP Addresses option is set to "no", these addresses are not processed in any special way - messages sent to servers with Client IP Addresses are processed in the same way as messages sent to servers with non-Client IP Addresses.

---

# Client-only Logins

| Logins from Non-Client IP Addresses |
|---|
| prohibit |
| allow |

If you do not plan to support mobile users, you may want to select the "prohibit" option for the Logins from from Non-Client IP Addresses setting, to allow any type of "login" operation from the Client IP Addresses only.
Connections from other addresses are accepted, but only the services that do not require "login" operations will be available: SMTP mail transfer, incoming SIP requests, HTTP access to File Storage, public Mailing List browsing, etc.

**Note:** Please check that your Client IP Addresses field is filled with your client addresses and read the Security section before you select this option.

---

# Relaying for Mobile (non-client) Users

If some of your users travel a lot, they may use various ISPs to connect to the Internet, and as a result they will connect to your Server from various IP addresses. If those users use your Server as the SMTP mail relay to which they submit all outgoing messages, Relay Restrictions will not allow them to send messages when their IP addresses are not in the Client IP Addresses list.

You should **not** select the "prohibit" for the Logins from Non-Client IP Addresses setting, if you want to support mobile users. Select the Allow option instead.

## The SMTP AUTH method

Most E-mail clients support "SMTP AUTH" - the standard SMTP Authentication method that allows a mailer to authenticate the user (the sender). If the SMTP module receives a message from an authenticated user, the message is marked as being "submitted from a local Account", and this message can be relayed to the Internet.

## The Read-then-Send method

To allow mobile users with older mailer applications (those not supporting SMTP AUTH) to send messages via the CommuniGate Pro server, the POP, IMAP, and other "access-type" modules check if an authenticated user has connected from an IP address not listed as one of the Client Addresses. During that POP/IMAP session, and for some time after the session is closed, that IP address is considered to be a "Client Address", so that users can send mail via your Server right AFTER they have checked their mail.

| Logins from Non-Client IP Addresses | | |
|---|---|---|
| prohibit | | |
| allow | | |
| Process as a Client IP Address for | 30 minutes | after the user disconnects |
| Remembered IP Addresses Limit: | 100 | |

The expiration time is used because of the "dynamic IP address" policies of most ISPs: when a user disconnects from an ISP modem pool, and some other user connects to the Internet via the same ISP, the same IP address can be assigned to that other user.

Inform your users about the expiration time. They should compose all their messages off-line, then they should connect to the Internet using any ISP, check their mail on your Server, and only then they can send the queued outgoing messages. If they want to reply to some messages they have just retrieved from the Mailbox on your Server, they should use the Get Mail command in their mailer application again, and only then can they send their replies.

Since many mailer applications try to send queued messages first, the SMTP module checks the Return-Path (the address in the `Mail From` SMTP protocol command). If that address is an address of a registered user, a to-be-relayed message is not rejected with the "permanent failure" error code. Instead, a "temporary failure" code is returned (with the "try to authenticate first" comment). Many mailers do not interrupt the mail session when they receive such a code, and continue by authenticating the user, retrieving the user mail, and retrying to send the queued messages. The queued messages will be accepted this time, because the user is authenticated from the same address.

An SMTP (message submit) session should start either during a POP or IMAP session, or within the expiration time after the end of the POP/IMAP session. Then that SMTP session can last as long as needed (several hours), if the queued messages are large and the link is slow.

## Account and Domain Settings

Support for mobile users can be disabled on per-account and per-domain basis by disabling the `Mobile` option in the Enabled Services section on the Account Settings and Domain Settings pages. If this service is disabled for an Account, the Account user will able to connect only from the internet addresses included into the Client IP Addresses list.

Mail relaying for mobile users can be disabled on per-account and per-domain basis by disabling the `Relay` option in the Enabled Services section on the Account Settings and Domain Settings pages. If an Account or a Domain has this service disabled, the IP address from which the user connects is not remembered as "a temporary client IP address", and the SMTP Authentication will not allow this user to relay messages via your SMTP module. This setup is useful when you give users Accounts on your Server, but you do not want them to be able to relay SMTP mail through your Server (they are forced to submit messages using the WebUser Interface or any other non-SMTP methods).

# Return-Path Address Verification

If your SMTP module can accept incoming TCP connections, your server can be used by spammers as a mail relay engine: they can distribute their messages all over the world using your server. To protect your site from spammers, the SMTP module can verify the Return-Path address (specified with the Mail From SMTP command) of incoming messages.

The SMTP module parses the message Return-Path (Mail From) address and rejects it if:

- the Return-Path domain name is an empty string (no domain specified)
- the Return-Path address is routed (via the Server Router) to the ERROR address

When the Verify HELO and Return-Path option is selected in the SMTP Service Settings, the SMTP module refuses to receive a message if:

- the Return-Path domain name is specified as an IP address, and that address is not included into the Client Addresses list

If the connection comes from an address not included into the Client IP Addresses list, additional DNS verification checks are done, and the SMTP module rejects the Return-Path address if:

- the Domain Name System does not have MX or A records for the Return-Path domain (an unregistered domain)
- the Domain Name System has an MX record for the Return-Path domain, but it points to an A-record that does not exist (a faked domain)

The SMTP module uses the Router after it parses the Mail From address. If that address is an address of a local user, or the address is known (rerouted) with the Router, the Mail From address is accepted. This eliminates Domain Name System calls for the addresses "known" to the Server.

The addresses routed to the ERROR address are rejected, so you can specify "bad" addresses and domains in the Router.

Examples:
    If you do not want to accept mail from any address in the offenderdomain.com domain, put the following line into the Router settings:

```
offenderdomain.com = error
```
    or
```
<*@offenderdomain.com> = error
```

    If you do not want to accept mail from all addresses starting with "promo" in the offenderdomain.com domain, put the following line into the Router settings:
```
<promo*@offenderdomain.com> = error
```

If the Return-Path domain cannot be verified because the Domain Name Server that keeps that domain records is not available, the module refuses to accept the message, but instead of a "permanent" error code the module returns a "temporary" error code to the sending system. The sending system will try again later.

You can tell the SMTP module to use SPF DNS records to check that messages with the specified Return-Path can come from the sender's network (IP) address.

You can tell the SMTP module to use the Reverse Connect method:

- the SMTP module makes a connection to the server that receives E-mail for this Return-Path address
- the SMTP module sends the Return-Path address to that server and checks the server response

If the server rejects this address, the SMTP module rejects the supplied Return-Path address, too.

# Blacklisting Offenders

Since your SMTP module can accept incoming TCP connections, your server can be used by spammers as a mail relay engine: they can try to distribute their messages all over the world using your server, and they can also send a lot of unwanted messages to your users.

To protect your system from known spammer sites, CommuniGate Pro provides several methods to maintain "black lists" of offending hosts IP addresses.

When a "blacklisted" host connects to your server and tries to submit a message via SMTP, it gets an error message from your SMTP module and mail from that host is not accepted.

**Note:** connections from "blacklisted" hosts are still accepted. If you want to reject all connections from the certain Network Addresses, see the [Denied Addresses](#) section.

Use the WebAdmin Interface to open the Network pages in the Settings realm, then open the Blacklisted IP Addresses page.

## Specifying Offender Addresses

Enter the IP addresses of offending hosts in the Blacklisted IP Addresses field:

| Blacklisted IP Addresses |
|---|
|  |

Each line can contain either one address:

    10.34.56.78

or an address range:

    10.34.50.01-10.34.59.99

A comment can be placed at the end of a line, separated with the semicolon (`;`) symbol. A line starting with the semicolon symbol is a comment line, and it is ignored.

## Using DNS-based Blacklisting (RBL)

It is difficult to keep the Server "blacklist" current. So-called RBL (Real-time Blackhole List) services can be used to check if an IP address is known as a source of spam.

Some ISPs have their own RBL servers running, but any RBL server known to have a decent blacklist can be used with your CommuniGate Pro server. Consult with your provider about the best RBL server available.

To use RBL servers, select the `Use Blacklisting DNS Servers` option and enter the exact domain name (*not* the IP address!) of the RBL server. Now, when the SMTP module accepts a connection from an IP address `aa.bb.cc.dd`, and this address is not listed in the Blacklisted, Unblacklistable, or Client Addresses lists, the module composes a fictitious domain name `dd.cc.bb.aa.rbl-server-name` where rbl-server-name is the domain name of the RBL server you have specified.

The SMTP module then tries to "resolve" this name into an IP address. If this operation succeeds and the retrieved IP address is in the `127.0.0.2-127.1.255.255` range, then the `aa.bb.cc.dd` address is considered to be blacklisted.

**Note:** this option results in an additional DNS (Domain Name System) operation and it can cause delays in incoming connection processing.

| Use Blacklisting DNS Servers (RBLs) |
|---|
|  |

You can specify several RBL Servers using the last (empty) field in the RBL Server table. To remove a server from the list, enter an

empty string into its field. The more servers you use, the larger the incoming connection processing delay. If you really need to use several RBL servers, but do no want those additional delays, make your own DNS server retrieve the RBL information from those servers (using daily zone updates) and use your own DNS server as an RBL server.

**Note:** An RBL server failure can cause very long delays for incoming connections. To avoid these situations, the requests to RBL servers are sent not more than twice, each time with the minimal time-out.

## Blacklisting Domains by Name

When a client connects from a network address not listed in the Blacklisted IP Addresses lists, and the Blacklist by DNS Name option is enabled, the server tries to get the domain name for that IP address (if the IP address is `aa.bb.cc.dd`, the Server tries to retrieve the PTR record for the `dd.cc.bb.aa.in-addr.arpa` name).
If the PTR domain name is retrieved, it is checked against the strings specified in the table (these strings can include the wildcard (`*`) symbols). If the retrieved name matches one of the table strings, the address is processed as a blacklisted one.

| Detect Blacklisted by DNS Name |
| --- |
| |

**Note:** if the Blacklist by DNS Name option is enabled, the server has to make an additional reverse-lookup DNS operation (unless the Detect Clients by DNS Name has been already enabled). This additional DNS operation can cause additional delays when processing incoming SMTP connections, so enable this option only when needed, and only when you cannot specify all blacklisted addresses explicitly - in the Blacklisted IP Addresses list.

**Note:** if the reverse-lookup DNS operation fails, the server places the DNR error code into the container used to keep the reverse-lookup DNS operation results (DNS names). The error code is enclosed in parenthesis. To blacklist all network addresses that do not have reverse-DNS records, place the `(host name is unknown)` string into the Blacklist by DNS Name table:

| Detect Blacklisted by DNS Name |
| --- |
| |

## Un-listing Addresses (White Hole Addresses)

When using RBL Servers or DNS Names for blacklisting, you may want to avoid blacklisting certain sites.

Enter those "unblacklistable" addresses using the same format you use for Blacklisted IP Address list:

| UnBlacklistable (White Hole) IP Addresses |
| --- |
| |

You can "unblacklist" addresses using their DNS (PTR) names:

| Detect White Holes by DNS Name |
| --- |
| |

```
|                                                                            |
|                                                                            |
```

Select the checkbox to enable this option and enter the DNS domain names you do not want to be blacklisted. This can be useful if some "good" addresses are blacklisted with the RBL services you use.

**Note:** The explicitly specified Blacklisted IP Addresses cannot be "unblacklisted" using the DNS Names.

## Processing Messages from Blacklisted Addresses

You can modify the SMTP module reaction on messages coming from blacklisted IP addresses. Instead of rejecting them (by adding the `@blacklisted` suffix to all their recipient addresses), the module can accept those message, but add a specified Header field to each of them:

| Messages from Blacklisted IP Addresses | |
|---|---|
| Reject | Add Header: |

The Header field string can contain the following macro combinations:

- `^0`. This combination is replaced with the name of the RBL host blacklisting the address. If no RBL host was used, the combination is replaced with an empty string.
- `^1`. This combination is replaced with the network address of the blacklisted host.

## Temporarily Blocked Addresses

CommuniGate Pro maintains its own "temporary Blacklist". Network addresses in that list are blocked for a certain time period only.

| Temporarily Blocked IP Addresses | | | |
|---|---|---|---|
| Block after: | 1000 | failed Logins in | 10 min |
| Block after: | 1000 | protocol errors in | 10 min |
| Blocking Time: | 0 sec | Blocked Addresses Limit: | 1000 |

IP Addresses can be blocked when the Server detects some activity from those addresses that can be qualified as suspicious:

- too many protocol errors in [SMTP receiving sessions](#) (the number of errors is configured in the SMTP module)
- addresses routed to the [spamtrap address](#) in SMTP receiving sessions
- too many failed authentication (Login) attempts from the same address
- too many protocol errors (misformed packets) in the packet-based protocols, such as [SIP](#).

`Block after`
> Use this setting to specify when a Network IP Address should be blocked:
>
> `failed Logins in`
>> maximum frequency of failed Login attempts: if there are more failed Login attempts from some IP Address during the specified period of time, this IP Address is blocked.
>
> `protocol errors in`
>> maximum frequency of protocol errors or misformed packets: if there are more misformed protocol packets from some IP Address during the specified period of time, this IP Address is blocked.

`Blocking Time`
> Use this setting to specify for how long an IP Address should be blocked.

`Blocked Addresses Limit`
> Use this setting to specify the maximum number of IP Addresses the Server can block.

**Note:** addresses included into the [White Hole Addresses](#) list are never placed into the Temporarily Blocked Addresses list.

**Note:** in a [Dynamic Cluster](#) environment only the Cluster-wide Temporarily Blocked Addresses settings are in effect.

# Checking Network Address Status

Your IP tables can become quite large, making it difficult to check if a particular network address is recognized by the Server as a Client one, or as a Blacklisted one.

Use the Test Address panel located on the Client IP Addresses and Blacklisted IP Addresses pages:

| Test Address: | [10.0.1.89](host1.lan) is Trusted |
|---|---|

Enter an IP address and click the Test button. The IP address status appears.

The status shows the IP address you have entered. It can have the `Local` prefix, if the address is the local address of your Server, or it can have the `LAN` prefix, if the address is included into [LAN IP Addresses](#) list.

The "reverse-resolved" name is displayed if the Server had to perform the "reverse-resolving" DNS operation to get the address status.

The address and optional name are followed by the address status:

`Trusted`
the IP address is a Client IP address.
`TempTrusted`
the IP address is processed as a Client IP address because some user (with the Relay Service enabled) has recently authenticated from that address.
`Blacklisted`
the IP address is blacklisted. If the address is blacklisted because it is included into some RBL, the name of that RBL server is displayed.
`Regular`
all other addresses.

# Spam Traps

You can protect your site from incoming spam by creating and advertising one or several "spam-trap" E-mail addresses. The CommuniGate Pro Router detects a special local address, `spamtrap`. If your server receives a message, and at least one of its recipients is `spamtrap@`*`yourhost`* or at least one of its recipients is routed to `spamtrap`, the Server rejects the entire message.

You may want to create one or several alias records for "nice-looking" fictitious E-mail addresses and route those addresses to `spamtrap`:

```
<misterX> = spamtrap
<johnsmith@subdomain.com> = spamtrap
```

Alternatively, you can create [Forwarders](#) pointing to the `spamtrap` address.

Then you should do your best to help these addresses (`misterX@yoursite.com`, `johnsmith@subdomain.com`) to get to the bulk mailing lists used by spammers. Since most of those lists are composed by robots scanning Web pages and Usenet newsgroups, place these fictitious addresses on Web pages and include them into the signatures used when you and your users post Usenet messages. To avoid confusion, make the fictitious E-mail addresses invisible for a human browsing your Web pages and/or attach a comment explaining the purpose of these addresses.

Many bulk mailing lists are sorted by the domain name, and as a result many spam messages come to your site addressed to several recipients. These recipients are the E-mail addresses in your domain(s) that became known to spammers. When the fictitious, "spam-trap" addresses make it to those databases, most of spam messages will have these addresses among the message recipients. This will allow the Server to reject the entire messages, and they will not be delivered to any real recipient on your site.

When at least one of the incoming message recipient addresses is routed to the `spamtrap` address, the entire message is rejected, and the IP Address of the sending server is placed into the [Temporarily Blocked Addresses](#) list, unless this IP Address is included into the [Client IP Addresses](#) or [White Hole Addresses](#) lists.

# Banning Mail by Header and Body Lines

You can specify a set of message Header and Body lines to be used to detect spam. When the server receives mail in the RFC822 format (via SMTP, RPOP, POP XTND XMIT, PIPE modules), it compares each received header and body line with the specified lists. If a message contains one of the specified lines, the message is rejected.

You can use the wildcard ('*', asterisk) symbols in the Banned Lines you specify. Usually you should not use them, since you are expected to compose the "banned" lists by copying header or body lines from the known spam messages.

Message lines are compared to the specified Banned lines in the **case-sensitive** mode.

Each Header line can include the end of line symbols if the header field was "wrapped".

If a message header or body is encoded (using MIME or UU encoding), the lines are **not** decoded before they are compared to the Banned line sets.

To specify the set of Banned Lines, open the Queue pages in the Settings realm of the WebAdmin Interface, and click the RFCReader link.

| Banned Header Lines |
|---|
|  |

| Banned Body Lines |
|---|
|  |

To add a new line, enter it in the empty field, and click the Update button.

To remove a line, delete it from its field, and click the Update button.

---

# Filtering Mail

When a message is received with the Server, a set of Server-Wide Rules is applied. These Rules can be used to detect unwanted messages and reject, discard, or redirect them.

For example, the following Rule can be used to reject all messages that have a missing `To:` header field:

| Data | Operation | Parameter |
|---|---|---|
| Header Field | is not | |
| **Action** | **Parameter** | |
| Reject with | | |

You can create various filtering rules using all features of CommuniGate Pro Automated Mail Processing, including external filter programs started with the `Execute` Rule Action.

---

# Relaying Rerouted Messages

Read this section if you need to provide special relaying features.

If you place an alias record into the Router table:

```
NoRelay:<user> = user@other.host
```

then all mail from strangers to that user will be rerouted to that other.host server. If that server address is not included into the Client IP Addresses list, these messages will be treated as messages "from a stranger to a stranger", and they will be rejected if the Relay for Clients Only option is switched on.

To enable relaying, use the `Relay:` prefix:

```
Relay:<user> = user@other.host
<user> = user@other.host
```

When an address is being converted with such a record, it gets a marker that allows the server to relay messages to that address. If an address is modified with a record that has the `NoRelay:` prefix, this marker is not set, but it is not reset either - if it has has been set with some other Router record (see the example below).

The same situation exists if you want to reroute all mail for a certain domain to a different host (for example, if you back up that host), and that host address is not included into the Client IP Addresses list.

```
Relay:clienthost.com = client1.com
Relay:<*@clienthost.com> = client1.com
```

When the address modified with the Router record is not a "simple address", i.e. it contains several routes, as in `user%host1@host2`, or `<@host2:user@host1>` - the `Relay:` prefix does not set the flag that allows message relaying. This is done because the host to which the rerouted message is relayed may "trust" all messages that come from your host, and relaying addresses with multiple routes would allow someone to relay messages to anybody through your host and that other host.

If the receiving server is well-protected, too, you may need a Router record that allows relaying of any address rerouted with that record. Use the `RelayAll:` prefix for those records:

```
RelayAll:<report-*@clienthost.com> = report-*@client1.com
```

Very often you do not want the Router records to be used for actual relaying - you provide them for your own clients only, to specify a special path for certain addresses/domains. For example, if you want mail to `bigprovdier.com` to be sent via a particular relay `relay3.com`, you should place the following record into the Router table:

```
NoRelay:bigprovdier.com = bigprovdier.com@relay3.com._via
```

Without the NoRelay prefix, any host on the Internet could send messages to bigprovdier.com via your Server. The NoRelay prefix tells the Router not to add marker to addresses in the bigprovdier.com domain, so only your own users (clients) can send mail to bigprovdier.com domain using your Server.

**Note:** you may have an alias record in your Router:

```
Relay:<joe> = joe5@bigprovdier.com
```

This record tells the server to reroute all mail addressed to joe@mydomain.com to joe5@bigprovdier.com. Since this record has the `Relay:` prefix, anybody in the world can send E-mail messages and Signals to joe@mydomain.com and they will be successfully relayed to the bigprovdier.com domain.
The `joe5@bigprovdier.com` address will be converted to `joe5%bigprovdier.com@relay3.com._via` and sent via relay3.com host: the second address transformation does not add the "can relay" marker, but it does not reset the "can relay" marker set during the first transformation:

| Operation Applied | Address | Marker |
|---|---|---|
| Received (Original) address | joe@mydomain.com | NO |
| Main Domain (mydomain.com) cut-off: | joe | NO |
| Router Record:<br>Relay:<joe> = joe5@bigprovdier.com | joe5@bigprovdier.com | YES |
| Router Record:<br>NoRelay:bigprovdier.com = bigprovdier.com@relay3.com._via | joe5%bigprovdier.com@relay3.com._via | YES |
| SMTP/SIP Module:<br>accepted for the host relay3.com | joe5@bigprovdier.com | YES |

# Cluster Setup

When a Server is a member of a [Dynamic Cluster](), the WebAdmin Network and Queue Settings pages provide links that allow you to switch between the local (server-wide) and the cluster-wide Settings.

The cluster-wide Address Tables (Client IP Addresses, Blacklisted IP Addresses, Unblacklistable IP Addresses) are processed as extensions of the server-wide tables: an address is considered to be listed if it is included into either the server-wide or into the cluster-wide table.

The cluster-wide "Client By DNS Name" list is processed as an extension of the individual server-wide list of "Client By DNS Name" domain names (if the Detect Clients by DNS Name option is enabled on the cluster-wide page).

The cluster-wide "Blacklist By DNS Name" list is processed as an extension of the individual server-wide list of "Blacklist By DNS Name" domain names (if the Blacklist by DNS Name option is enabled on the cluster-wide page).

The cluster-wide list of "Blacklisted" RBLs is processed as an extension of the individual server-wide RBL server lists. Each server will consult with the locally-specified RBL servers first, then it will consult with the RBL servers specified in the cluster-wide settings.

The cluster-wide "Banned" settings are processed as extensions of the server-wide settings: a message is banned if its header or body line is listed in the server-wide or in the cluster-wide settings.

**Sys Admin** **Logs** **Router** **Protection** **Security** **PKI** **Intercept** **Scalability** **Alerts** **Statistics**

# Security

- **Authentication Methods**
- **Account Passwords**
- **CommuniGate Passwords**
- **OS Passwords**
- **Kerberos Authentication**
    - **Integrating with Microsoft Active Directory**
- **Certificate Authentication**
- **External Authentication**
- **Account Name Harvesting and Password Attacks**
- **Granting Access Rights to Users**
- **Restricting Access**
- **Impersonating**
- **SessionID Authentication Method**
- **Access Control Lists (ACLs)**

The CommuniGate Pro Server ensures that only certain users are allowed to access certain resources.

The CommuniGate Pro Server can authenticate its users, and it can also reject connections from outside of the "client networks".

# Authentication Methods

The CommuniGate Pro Server supports both clear-text and secure SASL authentication methods for the following TCP-based session-oriented protocols:

- POP (as specified in RFC1734)
- IMAP (as specified in RFC2060)
- LDAP (as specified in RFC2251)
- ACAP (as specified in RFC2244)
- SMTP (as specified in RFC2554)
- FTP (as specified in RFC2228)
- XMPP (as specified in RFC3920)

These secure methods allow mail clients to send encrypted passwords over non-encrypted and insecure links. If anybody can monitor your network traffic, SASL methods ensure that the real passwords cannot be detected by watching the client-server network traffic.

As an alternative to SASL methods, secure links (SSL/TLS) can be used between the client mailer and the server. When an SSL link is established, the entire network traffic between the server and the client is encrypted, and passwords can be sent in clear text over these secure links.

You can force an Account user to use either a SASL authentication method or SSL/TLS links if you enable the `Secure Method Required` option in the Account Settings. When this option is enabled, the Server rejects all authentication requests that send passwords in the clear text format over insecure links.

The CommuniGate Pro Server supports the following insecure (clear text) SASL authentication methods:

- PLAIN

```
LOGIN
```

The CommuniGate Pro Server supports the following secure SASL authentication methods:

- `CRAM-MD5`
- `DIGEST-MD5`
- `GSSAPI`

The CommuniGate Pro Server supports the following GSSAPI authentication methods:

- `Kerberos V5`
- `NTLM`

The CommuniGate Pro Server supports the following SASL-EXTERNAL authentication methods:

- `TLS-Certificate`

The CommuniGate Pro Server supports the non-standard `NTLM` and `MSN` SASL methods used in Microsoft® products.

The CommuniGate Pro supports the secure `APOP` authentication method (used mostly for the POP protocol), and the insecure "regular login" method for the protocols that support Clear Text Login.

The CommuniGate Pro Server supports the special `SessionID` Authentication method.

Use the WebAdmin Interface to open a Domain Settings page and find the Login Methods panel:

| Login Methods | | | Help |
|---|---|---|---|
| CLRTXT | CRAM-MD5 | DIGEST-MD5 | APOP |
| GSSAPI | NTLM | MSN | SESSIONID |

CLRTXT
  When this option is selected, the Server advertises all supported non-secure (clear text) authentication methods for this Domain.

CRAM-MD5, DIGEST-MD5
  When these options are selected, the Server advertises the secure CRAM-MD5 and DIGEST-MD5 authentication methods for this Domain.
  Do not select these options if the Domain Accounts use one-way encrypted passwords, OS Passwords, or other authentication methods that do not support secure authentication methods.

APOP
  When this option is selected, the Server provides a special initial prompt for POP and PWD connections. Mail clients can use this prompt to employ the secure APOP authentication method.
  Do not select this option if the Domain Accounts use one-way encrypted passwords, OS Passwords, or other authentication methods that do not support secure authentication methods.

GSSAPI
  When this option is selected, the Server advertises support for the GSSAPI authentication method.
  Do not select this option if the Domain is not set to support GSSAPI methods (for example, you have not specified the required Kerberos Keys).

MSN, NTLM
  When these options are selected, the Server advertises the non-standard MSN and NTLM authentication methods (used in some Microsoft products) for this Domain.
  Do not select these options if the Domain Accounts use one-way encrypted passwords, OS Passwords, or other authentication methods that do not support secure authentication methods.
  **Note:** The Microsoft Outlook products for various versions of MacOS do not work correctly with the MSN method if more that one account is configured in those products.
  **Note:** Some Microsoft products send incorrect credentials when they detect that the server supports the NTLM SASL method. While those products then resend the correct credentials, the failed login attempts produce Failure-level Log records and may increase the "failed logins" counter too quickly, so the account becomes "temporarily locked".

The Advertise options control only the session-type services (SMTP, POP, IMAP, ACAP, PWD, FTP), and they do not have any

effect on the transaction-type services (HTTP, SIP).

The Advertise options control only how the methods are advertised. Client applications can still use all these methods, even if these options are switched off.

SESSIONID
> This option enables the SessionID Authentication method for the Domain.

# Account Passwords

The CommuniGate Pro Server supports several passwords for each account.

One password is the CommuniGate Pro's "own password ". This password is stored as an element of the Account Settings, and it can be used with the CommuniGate Pro Server only.

Additional variants of CommuniGate Pro internal password can be specified with *tag* names. When authenticating with these *tagged>* passwords the authenticating client application should specify the password tag after account name separated with $ symbol: *user$tag*.

The other password is the "OS password". The user may be registered with the Server OS and the CommuniGate Pro Server can check the supplied password against the password set in the Server OS registration information for this user.

An account can have the Authentication URI specified that is used to authenticate to an external LDAP server. This method works only with Clear Text authentication methods.

An account can have the External Password option enabled. In this case, user authentication is done using any custom authentication program running as a separate process (see below).

The system administrator can enable any set of passwords for any user account. On larger sites, it is better to enable these options using the Server-wide or Domain-wide Default Account Settings.

When several passwords are enabled for an account, the Server first checks the CommuniGate (internal) password, then the OS password, then Authentication URI if not empty, and then tries to use the External Authentication program. If at least one of these passwords matches the password presented with the client application, the application is granted access to that account.

# CommuniGate Passwords

CommuniGate passwords are strings stored in the Account Settings. Password strings can be stored in the clear-text format or in encoded format. The Password Encryption Account Setting specifies the encryption to use when the account password is updated. When this setting is changed, the currently stored password is not re-encrypted.

When the U-crpt Password Encryption option is selected, the CommuniGate passwords are stored using the standard Unix crypt routine. If the UB-crpt Password Encryption option is selected, an enhanced Blowfish-based encryption is used. U-crpt and UB-crpt methods implement a one-way encryption. As a result, the Server cannot decrypt them into their original (clear text) form, and it cannot use them for secure (SASL) Authentication Methods. Use these encryption methods only if you need compatibility with legacy password strings, but cannot use the OS passwords - it is usually more important to support "on-the-wire" security (using SASL methods), rather then "on-the-disk" security (using one-way password encryption methods).

U-crpt passwords can contain special prefixes. These prefixes allow you to import passwords encrypted using other password encryption methods.
See the Migration section for more details.

**Note:** please remember that the plain Unix crypt routine uses only the first 8 symbols of the password string.

If the CommuniGate Password is absent or empty, it cannot be used to log into the Account even if the Use CommuniGate Password option is enabled. But if the user has logged in using the OS Password or the External Authentication method, the user can specify (update) the Account CommuniGate Password. This feature can be used to migrate users from legacy mail systems where you can not compose the list of accounts with their non-encrypted (plain text) user passwords.

# OS Passwords

When the Server checks the OS password, it composes the *username* string using the Account [OS User Name setting](#). When the default setting `*` is used, the composed OS user name is the same as the Account name. By changing the OS User Name settings you can use different OS usernames for Accounts in different CommuniGate Pro domains.

| Server Operating System | Notes about OS Passwords |
|---|---|
| Microsoft Windows 95/98/ME | OS does not support passwords, the Use OS Password option does not work. |
| Microsoft Windows 200x/XP/NT/Vista | The Windows NT domain authentication system is used. The Windows account used to run the CommuniGate Pro Messaging Server should have the `Act as part of the operating system` privilege.<br><br>The --BatchLogon [command line option](#) can be used to tell the Server to use the LOGON_BATCH authentication method (if the option is not present, the LOGON_NETWORK method is used).<br><br>The Server checks if the composed OS user name contains the percent (`%`) symbol. If the symbol is found, the part of the name before that symbol is used as the Windows account name, and the part after that symbol is used as the Windows domain name.<br>If Accounts in the `company1.dom` CommuniGate Pro domain have the OS User Name setting set to `*%comp1`, then the OS user name for the CommuniGate Pro Account `joe` will be `joe%comp1`, and the CommuniGate Pro Server will use the Windows `LogonUser` API to try to authenticate the mail client as the Windows user joe in the Windows domain comp1. |
| Unix-based systems | The `passwd` and `shadow` or other OS-supported authentication mechanisms are used. |
| OS/400 systems | The `user profile` authentication mechanisms are used. |
| OpenVMS systems | The supplied user name and password strings are converted to uppercase, and then the OpenVMS authentication mechanisms are used. |
| BeOS | OS does not support passwords, the Use OS Password option does not work. |

The OS passwords are one-way-encrypted passwords. As a result, they cannot be used for [Secure Authentication Methods](#).

---

# Kerberos Authentication

The CommuniGate Pro Server supports the Kerberos authentication method. The Kerberos method is based on the "tickets" that client applications send to the server. These tickets are issued by Kerberos authorities (Key Distribution Centers, KDC) that share a common "key" with the Server. See the Kerberos documentation for the details.

To support Kerberos Authentication, you need to add Kerberos Server key(s) to the CommuniGate Pro Server, on the per-domain basis. Create a server "principal" in your KDC database. The principal name should be equal to the name of CommuniGate Pro Domain or one of its Domain Aliases. Export the created key as a `keytab` file.

Open the Domain Settings using the CommuniGate Pro WebAdmin Interface, and follow the Security and Kerberos links. The list of Domain Kerberos Keys will be displayed:

| Realm | | Issued | Version | Type |
|---|---|---|---|---|
| REALM1.COM | (3)imap/domain1.com | 19-05-2004 17:36:33 | 6 | RC4-HMAC |
| REALM1.COM | (3)imap/domain1.com | 21-05-2004 17:32:35 | 9 | DES-MD5 |
| REALM1.COM | (3)imap/domain1.com | 24-05-2004 12:41:55 | 10 | DES-MD5 |
| REALM1.COM | (3)imap/domain1.com | 24-05-2004 17:33:27 | 13 | DES-CRC32 |
| **no file selected** | | | | |

Each Domain can have several Kerberos Keys. To add Keys, click the browser file-select button and select the `keytab` file exported from KDC. Click the Import Keys button to add keys from the file to the set of the Domain Kerberos Keys.

To remove Keys, mark the Keys using checkboxes and click the Delete Marked button.

Domain Administrators can Add or Remove Kerberos Keys only if they have the [KerberosKeys Access Right](#).

When the Server receives a Kerberos Ticket, it extracts the Server Name ("sname") from the Ticket. If the Server Name has only 1 component (`domain.dom`), this component is used as the target Domain name (*ticket-domain-name*). If the Server Name has 2 or more components (`service/domain.dom`), then the second component is used. The Server then builds a fictitious E-mail address `LoginPage@ticket-domain-name` and tries to route this address. This is the same routing mechanism as one used for finding the target Domain for [HTTP](#) requests.

If the target Domain is found, the Server looks for the proper key in the list of the Kerberos Keys for that Domain. If the Key is found, and the Ticket and Authorization info can be decrypted with that Key, the user is authenticated. The name of the Account is taken from the Client Name specified in the Ticket. That name must be a "simple" name, i.e. it cannot contain `@` or `%` symbols.

CommuniGate Pro adds the name of the target Domain to the retrieved user name and uses the resulting E-mail address as the name of the Account to open.

**Note:** after the resulting user name is processed with the Router, it checks that the target Account belongs to the same Domain as the Domain the Kerberos Key was retrieved from, so Administrators of one Domain cannot create Kerberos tickets allowing users to access Accounts in other CommuniGate Pro Domains.

Only Accounts with enabled Kerberos Authentication method can be accessed using Kerberos tickets.

## Integrating with Microsoft Active Directory

You may want to use Microsoft Active Directory as your Kerberos Key Distribution Center (KDC). Follow these steps:

- create an Active Directory account `cgatepro` (you may want to use a different name)
- use the Microsoft `ktpass` utility to export keys:

```
ktpass -princ service/domainName@REALMNAME -mapuser cgatepro -pass password -out keytab.data -crypto
DES-CBC-MD5 -ptype KRB5_NT_SRV_HST
```

  where

  *service*
  > is the name of the CommuniGate Pro service you want to use: `imap` for IMAP and MAPI, `HTTP` for Web browsers
  *domainName*
  > is the name of the CommuniGate Pro Domain your Kerberos users should log into
  *REALMNAME*
  > is the name of the Kerberos Realm - the Active Directory domain name with which you want to authenticate
  *password*
  > is the password to assign to the `cgatepro` account.


  Note: because of the bugs in the Windows 2000 Active Directory, it is recommended to use the `-kvno 0` parameter when using the `ktpass` command on that platform.
- Import the resulting keytab.data file into the CommuniGate Pro Domain Kerberos settings, as specified above.

See the Microsoft Knowledge Base article Q324144 for more details.

If you want to use Kerberos Authentication (Single Sign-on) with Microsoft browsers, please read the "HTTP-Based Cross-Platform Authentication via the Negotiate Protocol" article in the Microsoft documentation set (MSDN).

---

# Certificate Authentication

The CommuniGate Pro Server supports the Client Certificate authentication method. This method can be used when clients connect to the Server via secure [SSL/TLS](#) connections. The Server may request a client to send a Client Certificate (installed on the client computer), signed with the Trusted Certificate selected on the Server for the target Domain.

If a client sends such a Certificate, the E-mail address specified in the Certificate `subjectAltName` element (if any) or in the `Subject`

element E-mail field can be used for the Certificate-based Authentication. It is is interpreted as the name of the CommuniGate Pro Account to log into.

**Note:** after the supplied E-mail address is processed with the Router, it checks that the target Account belongs to the same Domain as the Domain the used to verify the supplied Certificate, so Administrators of one Domain cannot create Certificates allowing users to access Accounts in other CommuniGate Pro Domains.

Only Accounts with enabled Certificate Authentication method can be accessed using Client Certificates.

See the PKI section for more details.

# External Authentication

The CommuniGate Pro Server can use an external Helper program for user authentication. That program should be created by your own technical staff and it can implement authentication mechanisms required at your site but not supported directly with the CommuniGate Pro Server.

The External Authenticator can also be used:

- to automatically provision Accounts based on some external data source
- to assist with the Router operations
- to assist with Account provisioning.

The External Authenticator program name and its optional parameters should be specified using the WebAdmin Helpers page. Open the General page in the Settings realm, and click the Helpers link:

| External Authentication | | | |
|---|---|---|---|
| Log Level: | Major & Failures | Program Path: | |
| Time-out: | disabled | Auto-Restart: | 15 seconds |

See the Helper Programs section to learn about these options.

The External Authentication module System Log records are marked with the `EXTAUTH` tag.

If the External Authentication program is not running, all External Authentication requests are rejected.

To create your own External Authentication program, see the Helper Applications section to learn about the External Authentication interface protocol.

Sample External Authentication programs and scripts can be found at the Authentication Helpers site.

# Account Name Harvesting

Some spammers use 'brute force' attacks on mail systems, sending random names and passwords to system POP, IMAP, and other access ports. If the system sends different error messages for the "unknown account" and "incorrect password" situations, the attacker can harvest a large portion of the system Account names and then use those names for spam mailings.

Use the WebAdmin Interface to configure the Login Security options. Open the General pages in the Settings realm, and find the Login Security panel on the Others page:

| Login Security |
|---|
| Hide 'Account Unknown' messages |

Hide Unknown Account messages
    If this option is enabled, the Server does not send the `Unknown Account` and `Incorrect Password` error messages. Instead, both messages are replaced with the `Incorrect Account Name or Password` error message.

The CommuniGate Pro Server can temporarily disable all types of login operation for an Account that has seen too many incorrect login attempts. The [Account Settings](#) specify a time period and the number of incorrect login attempts that a user or users can make before the Account is disabled for login operations. The Account is re-enabled after the same period of time.

# Granting Access Rights to Users

In order to control, monitor, and maintain the CommuniGate Pro Server, one Postmaster Account is usually enough. But you may want to allow other users to connect to the administrator interfaces of your CommuniGate Pro Server: for example, you may want to allow an operator to monitor the Logs, but you do not want to grant that operator all Postmaster access rights.

You should be logged in as the Postmaster, or other Account with the "Can Do Everything" right in order to assign Access Rights.

To grant access rights to a user and/or to revoke those rights, open that user Account [(the Account Setting page)](#), and click the Access Rights link. The Access Rights page will appear.

The page lists all Access Rights and the rights granted to the selected user are marked.

The following access rights can be granted only to the users (accounts) in the Main Domain:

Can Do Everything (the Master right)
    If a user is granted this right, it has all access rights to all System components.
Can Modify Server and Module Settings (the Settings right)
    This right allows a user to change configuration of all CommuniGate Pro components and modules (SMTP, POP, Router, etc.)
Can Modify Directory Settings (the Directory right)
    This right allows a user to change configuration of the System [Directory](#)
Can Modify All Domains and Accounts Settings (the All Domains right)
    This setting specifies if the user is allowed to create, rename and delete Domains, Accounts and other Objects, and to change Domain, Account, and other Object Settings.
Can Read All Domains and Accounts Settings (the All Domains Read right)
    This setting specifies if the user is allowed to read Domain, Account, and other Object Settings.
Can Monitor (the Monitor right)
    This setting specifies if the user is allowed to view Server Logs, to monitor Server Queues, etc.

The following access rights can be granted to users in any domain:

Can Modify This Domain and its Account Settings:
    This setting specifies if the user is allowed to create, remove and delete Accounts within its own domain, and to change some of the Domain Settings. You usually assign this right to a user ("domain master") who will manage the Domain.

Initially, the user Postmaster in the Main Domain has the Master Access right.

Select the desired Access Rights and click the Update button.

The Access Rights are stored in a single file for each Domain, the Access.settings file stored in the Settings subdirectory of the Domain file directory. This makes it easy to check who is granted the Server Administration rights.

# Restricting Access

If you do not plan to support mobile users, you may want to restrict access to the Server accounts. Use the following option on the [Client IP Addresses](#) page:

Logins from Non-Client Addresses
    When the "prohibit" option is selected, all "login" operations (needed for POP, IMAP, SIP, XMPP, WebUser Interface, ACAP, PWD, etc.) are accepted only from the Server computer itself, and from the [Client IP Addresses](#).

    When an access module accepts a connection from an unlisted network address, and this option is selected, the module sends an error code to the client application, and the connection is closed immediately. Links with the rest of the Internet will be used only for mail [Transfer](#), Real-Time [Signals](#) exchange, and for HTTP access to Account [File Storage](#).

    When this option is selected, the SMTP AUTH operation can be used only if a client mailer or server connects from the network

address included into Client Addresses list.

When this option is selected, any Signaling operation that requires authentication can be used only if a client device or server connects from the network address included into Client Addresses list.
**Note: Before you enable this option, make sure that the network address you are currently using is included into the Client Addresses list: otherwise you will immediately lose HTTP Admin access to the Server.**

You can also specify the access restrictions on the lower (TCP) connection level. For each service (module), open the Listener page and specify the addresses the service (module) should or should not accept connections from. If a connection comes from an address that is not included into the `Grant` list or is included into the `Deny` list, the connection is closed immediately, and no module-level operations are performed.

# Impersonating

The CommuniGate Pro Server supports impersonating - a login mode when the credentials are supplied for one Account (the Authentication Account), while a different Account (the Authorization Account) is being opened.
It can also be used for Real-Time Registration.

Impersonating is supported for `PLAIN` and `GSSAPI` Authentication methods.

When Impersonating is used, the Server checks if the authentication Account credentials are valid, and if the requested service is allowed for that Account. It also checks if the Authentication Account has the `CanImpersonate` Domain access right.

# The SessionID Authentication Method

The CommuniGate Pro Server supports the special `SessionID` Authentication method. This method uses the session ID of a WebUser or XIMSS session instead of the Account password.
The method is useful for CGI programs and scripts.
This method is disabled by default (see above).

The method is a SASL method and requires "immediate" parameters in the authentication protocol command. The first parameter is the Account name, the second parameter, separated with the space symbol, is the session ID.
The SESSIONID authentication operation for the PWD module is:

```
AUTH SESSIONID userName session-ID
```

The SESSIONID authentication operation for the IMAP module is:

```
AUTHENTICATE SESSIONID bindata
```
where *bindata* are base64-encoded parameter data:

```
userName session-ID
```

If the user `john@doe.dom` has an open WebUser Session with the 114-bXaKw92JK1pZVB5taj1r ID, then the PWD command:
AUTH SESSIONID john@doe.dom 114-bXaKw92JK1pZVB5taj1r
opens the john@doe.dom account in this PWD session.

# Access Control Lists (ACLs)

The Account owner can grant certain rights to other users: a right to access certain Mailboxes, a right to control telephony features, etc.

Each element of the Access Control List contains a name and a set of access rights granted to that name.

An ACL element name can be:

`null@null`

This ACL element specifies the access rights granted to "guests" (non-authenticated users).

`anyone`
This ACL element specifies the access rights granted to everybody (all authenticated accounts).

`anyone@`
This ACL element specifies the access rights granted to all Accounts in the same CommuniGate Pro Domain.

`anyone@`*domainName*
This ACL element specifies the access rights granted to all Accounts in the CommuniGate Pro *domainName* Domain. The *domainName* should be the real Domain name, and not a Domain Alias name.

`accountName`
This ACL element specifies the access rights granted to the *accountName* Account user in the same CommuniGate Pro Domain. The *accountName* should be a real Account name, and not an Account Alias or a Forwarder.

`accountName@`*domainName*
This ACL element specifies the access rights granted to an Account user in a different CommuniGate Pro Domain. The *domainName* should be the real Domain name, and not a Domain Alias name.

`#`*groupName*
This ACL element specifies the access rights granted to all members of the *groupName* [Group](#) (in the same Domain).

`#`*groupName@domainName*
This ACL element specifies the access rights granted to all members of the *groupName* [Group](#) in a different CommuniGate Pro Domain. The *domainName* should be the real Domain name, and not a Domain Alias name.

An ACL element name can have a `+` or a `-` prefix.

Account owners always have all access rights to all objects (Mailboxes, functions) in their own Accounts.

For any other `someaccount` Account, the effective access rights are checked.

The effective access rights are calculated in several steps:

- If there is an ACL element for the `someaccount` name (without a `+` or a `-` prefix), then the Access right specified in that ACL element are used as the effective access rights.
  Otherwise
- All ACL elements without a `+` or a `-` prefix and matching the `someaccount` name are merged to form the "direct" access rights.
- All ACL elements with the `-` prefix and matching the `someaccount` name are merged to form the "removed" access rights.
- All ACL elements with the `+` prefix and matching the `someaccount` name are merged to form the "added" access rights.
- The "removed" access rights are removed from the "direct" access rights.
- The "added" access rights are merged with the "direct" access rights.
- The resulting "direct" access rights are used as the effective access rights.

When granting access rights, the real Account names, not Account Aliases should be used. If an Account `j.smith` has two aliases `john.smith` and `jonny`, the access rights should be granted to the name `j.smith`.

Examples:
Grant the Lookup, Select, and Seen access rights to all users in the same Domain, excluding the user John, who should have only the Lookup right, and the user Susan who should have the Lookup, Select, Seen, and Delete rights:

| | |
|---|---|
| `anyone@` | Lookup, Select, Seen |
| `-john` | Select, Seen |
| `+susan` | Delete |

Grant the Lookup, Select, and Seen access rights to all users in a different company2.com Domain, excluding the user john@company2.com who should have no access rights, and grant the Lookup, Select, and Delete rights to the user susan in a yet another company3.com Domain.

| | |
|---|---|
| `anyone@company2.com` | Lookup, Select, Seen |
| `-john@company2.com` | Lookup, Select, Seen |
| `susan@company3.com` | Lookup, Select, Delete |

The Access Control Lists can be set and modified using the [WebUser Interface](#), a [XIMSS](#), a [MAPI](#), or a decent [IMAP](#) client.

Introduction

Installation

SysAdmin

Network

Objects

Storage

E-mail

Real-Time

Access

Services

Directory

Clusters

Applications

Miscellaneous

Licensing

WebMail

Pronto!

PBX

Sys
Admin Logs Router Protection Security PKI Intercept Scalability Alerts Statistics

# Public Key Infrastructure

- **PKI Terminology**
- **Domain PKI Settings**
  - **Assigning a Private Key**
  - **Assigning a Certificate**
  - **Assigning a Certificate Authority Chain**
- **Using Self-Signed Certificates**
- **Trusted Root Certificates**
- **SSL/TLS Secure Connections**
- **Client Certificates**
- **S/MIME Functionality**
- **Domain S/MIME Settings**
- **Automatic S/MIME Encryption**
- **Stored Messages Encryption**
- **DKIM Message Signing**

Regular ("classical") cryptography methods use data blocks called "secret keys". Information encrypted using some "secret key" can be decrypted by anyone who knows the encryption method and posses the same "secret key". This type of cryptography is called symmetric cryptography.

An alternative cryptography method is based on key pairs - a "private key" and a "public key". These keys must be generated together using special algorithms. Any information encrypted with the "private key" can be decrypted by anyone who knows the matching "public key", and any information encrypted with the "public key" can be decrypted using the matching "private key".
The Public Key Infrastructure (PKI) is the technology based on this asymmetric cryptography.

# PKI Terminology

**Private Key**
A block of data (a large binary number) generated using one of the PKI algorithms. Each party taking part in secure communications should keep its Private Key securely. This key should never be transferred between communication parties.

**Public Key**
A block of data (a large binary number) generated together with the Private Key. Each party taking part in secure communications can and should distribute its Public key openly. It is assumed that Public Key can be learned by anyone, including hostile entities. Public Keys are usually distributed in the form of Certificates.

**Data Digest**
A relatively small block of data calculated by applying a special digest function to the original (usually larger) data block.

**Data Signature**
A Digest of the Data block encrypted using the Private Key of the Signer.

**Signed Data**
A data block with attached Signature of that block. A party receiving Signed Data can verify that the data block has not been modified in transit by using the Public Key of the Signer to decrypt the Signature and to compare the resulting Data Digest with the Data Digest it calculated itself.

**Certificate**
A data block with containing the name of the Certificate owner (called Certificate Subject), the Public Key of the owner, the

name of the Certificate Issuer, the serial number of the Certificate, and some additional data elements. This data block is signed by the Issuer.

Certificates play the role of Digital ID cards.

**Issuer**

A party that issues Certificates for other parties, signing them with the Private Key of the Issuer. Issuers are also called Certificate Authorities. Each certificate generated by a certain Issuer has a unique serial number.

**Trusted Authorities**

A list individually maintained by a communication party. Each list element contains the name of a "trusted authority" and its Public Key.

When a party receives any Certificate, it can check if the Certificate Issuer is included into the "trusted authority" list, and that the Certificate Signature can be verified using that "trusted authority" Public Key.

Modern operating systems allow users to securely maintain Trusted Authorities databases on their desktops.

**Root Authorities**

Globally recognized Certificate Authorities. Most modern operating systems pre-insert several Root Authorities into the client Trusted Authorities databases, making Root Authorities trusted by all client computers running these operating systems.

**Authority Chain**

A set of Issuer Certificates for a certain Certificate.

Some Authority X may be not widely accepted as a "trusted one", but its Certificate may be issued by a more widely trusted Authority Y. In this case Certificates issued by X won't be widely accepted, but if those Certificates are sent together with the X own Certificate, issued by Y, then these Certificates may be accepted by all parties that trust Y.

**Self-Signed Certificate**

A Certificate issued by a party for itself. The Subject and Issuer of such a Certificate are the same. The Self-Signed Certificate contains the party Public Key and is signed using the Private Key of the same party.

Self-Signed Certificates can be trusted only if other parties explicitly include them into their lists of "trusted authorities".

**Multiparty Encryption**

An encryption method used to send data to parties with known Certificates. A single encrypted messages can be independently decrypted by any party that possesses a Private Key matching one of the Certificates used for encryption.

---

# Domain PKI Settings

Each CommuniGate Pro Domain has its own PKI settings. They include a Private Key associated with the Domain and Certificates containing the matching Public Keys.

To configure the Domain PKI settings, use the WebAdmin Interface to open the Domain Settings page for the target Domain, and click the Security link. The PKI page will appear:

| PKI Services |
| --- |
| default(Test) |
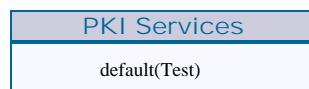
This option allows you to specify the PKI Mode for this Domain:

`Disabled`

If this option is selected, PKI Functions for this Domain are disabled. If this option is selected, all other Domain PKI settings have no effect.

`Test`

If this option is selected, the Server-wide Test Private key and Test Certificate are used for this Domain. You do not have to configure any other Domain PKI settings if this option is selected. Use this mode for testing purposes only.

The Server-wide Test Certificate uses the Server Main Domain name as its Subject and `CommuniGate Systems, Inc.` as the Issuer. The Test Certificate expires 30 days after the last Server restart time.

`Enabled`

If this option is selected, the Domain PKI functions are enabled.

## Assigning a Private Key

Initially CommuniGate Pro Domains do not have any Private Keys assigned. You should select the size of the key and click the Generate Key button to create a random Private Key and assign it to the Domain.

| Private Key |
|---|
| Size: 512 |

**Note:** depending on your server hardware platform, it can take up to several seconds to generate a 2048-bit Key.

Only after you assign a Private Key, the Certificate-related fields will appear on the Security page.

You can use any third-party program to generate a Private Key. You should instruct that program to output the Private Key in the PEM format (as shown below).
Select the Import option in the Size: menu and click the Generate Key button. A text field appears. Copy the PEM-encoded Key (in the RSA or PKCS#8 format) into that text field, and click the Generate Key button:

| Private Key |
|---|
| Size: Import |
| Enter a PEM-encoded Private Key: |

**Note:** Make sure that the key you import is not password-encrypted. Something like the following starting lines:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-CBC,90C96A721C4E4B0B

GzLyio+Or3zXm1N7ILWlYDsR6cgPlzHomAxi6aeUthl4lSqBHaqMlh+/76I/6sNx
................
```

indicate that the Private Key is encrypted and it cannot be imported into the Server.

If the Private Key is set correctly, and the Key can be used for public/private key cryptography, you will see the following panel:

| Private Key | |
|---|---|
| Key Size: | 512 bit |
| Encryption Test: | *Verification String is OK* |

If the Key Test field indicates an error, the imported Private Key cannot be used for public/private key cryptography.

Use the Remove button if you want to remove the entered Domain Private Key. Since the Domain Certificate can be used with one and only one Private Key, it becomes useless when you delete the Private Key, so the existing Domain Certificate will be removed, too.

## Assigning a Certificate

A Domain must have a Certificate to support PKI functions.

The Domain Certificate name (the Common Name part of the Certificate Subject field) should match the domain name used by client applications.
If a CommuniGate Pro Domain has Domain Aliases, attempts to connect to the Server using a Domain Alias name will result in warning messages on the client workstations notifying users about the name mismatch. Since a Certificate can contain only one name, select the name (the real Domain name or one of the Domain Aliases) that your users will use in their client application settings. If your CommuniGate Pro Domain name is company.dom, and that domain name does not have a DNS A-record, but the Domain has an Alias mail.company.dom that has an A-record pointing to the CommuniGate Pro Server, your users will use the mail.company.dom name in their client settings and WebUser Interface URLs, so the Domain Certificate should be issued for the mail.company.dom name rather than the company.dom name.

You may also use "wildcard" domain names for your certificates. If the Domain name has at least 2 components, the Common Name menu will contain a "wildcard" domain name: the name of the Domain with the first component substituted with the asterisk (*) symbol. If the Domain name has only 2 components, the asterisk component is added to form a 3-component name.

To create a Certificate, fill the fields in the Certificate Attributes table:

| Certificate Generator | |
| --- | --- |
| Common Name: | d1.communigate.com |
| Country: | |
| Province: | |
| City: | |
| Organization: | |
| Unit: | |
| Contact: | |

**Common Name**
> When the Certificate is sent to a client application, the application checks that the Certificate Common Name matches the name the user has specified in the URL and/or in the mailer settings.

**Contact**
> This field must contain a valid E-mail address, though that address does not have to be inside this CommuniGate Pro Domain.

All other fields are optional.

To receive a Certificate from an external source ("trusted authority"), click the Generate Signing Request button. A text field containing the PEM-encoded CSR (Certificate Signing Request) will appear:

| Certificate Generator | |
| --- | --- |
| Common Name: | d1.communigate.com |
| Country: | |
| Province: | |
| City: | |
| Organization: | |
| Unit: | |
| Contact: | |
| **Enter a PEM-encoded Certificate** | |
| | |
| *submit this request to a Certification Authority and paste the result below* | |

<table>
<tr><td colspan="2" align="center">**Enter a PEM-encoded Certificate**</td></tr>
<tr><td colspan="2" style="height:350px;"></td></tr>
</table>

Copy the CSR text and submit it to the Certification Authority (CA) of your choice. You can submit via E-mail or using a Web form on the CA site. The Certification Authority should send you back the signed Certificate in the PEM-format. Enter that Certificate into the bottom field and click the Set Certificate button.

If the Certificate is accepted, the Certificate information is displayed:

| Domain Certificate | | |
|---|---|---|
| **Issued to:** | Country | US |
| | Province | CA |
| | City | Sausalito |
| | Organization | ACME Yacht Rentals, Inc. |
| | Unit | On-line Services |
| | *Common Name* | d1.communigate.com |
| | Contact | bill@domain.company |
| **Issued by:** | Country | US |
| | Province | CA |
| | City | Sausalito |
| | Organization | ACME Yacht Rentals, Inc. |
| | Unit | On-line Services |
| | *Common Name* | d1.communigate.com |
| | Contact | bill@domain.company |
| **Serial Number:** | 89AB673940123456 | |
| **Valid:** | From: 06-Nov-07 | Till: 05-Nov-09 |

The Certificate panel shows the Certificate Issuer (the Certificate Authority), the Certificate Subject (the data you have entered and the selected domain name), the Certificate serial number and the validity period of this Certificate.

**Note:** the entered Private Key and Certificate will be used for the Domain secure communications ONLY if the PKI Services option is set to `Enabled`.

**Note:** the Certificate contains the Domain name or a Domain Alias as a part of the "Subject" data.
When you rename the CommuniGate Pro Domain, the domain name in the Domain Certificate does not change, and the client applications may start to warn users about the name mismatch.

Click the Remove Certificate button to remove the Domain Certificate.

## Assigning a Certificate Authority Chain

If the Certificate issuer is known to the users client software (mailers and browsers), the warning message does not appear on the user screen when the client software receives a Certificate from the Server. In many cases, the "trusted authority" does not issue certificates itself. Instead, it delegates the right to issue certificates to some other, intermediate authority. When your Server uses a Certificate issued by such an authority, the Server should also present the Certificate of that authority issued by the "trusted authority". The client software would check your Certificate first, then it will detect that the issuer of your Certificate is not a "trusted authority" and it will check the additional Certificate(s) the Server has sent. If that additional Certificate is issued by a "trusted authority", and it certifies the issuer of your Domain Certificate, your Certificate is accepted without a warning.

When you receive a Certificate from a Certificate Authority that is not listed as a "trusted authority" in the client software settings, that intermediate Certificate Authority (CA) should also give you its own Certificate signed with a "trusted authority". That Certificate should be in the same PEM format as your Domain Certificate:

| Certificate Authority Chain (Optional) |
|---|
| |

The CA Chain may include several certificates: the first one certifies the issuer of the Domain Certificate you have entered, but it itself may be issued by some intermediate authority. The next Certificate certifies that intermediate authority, etc. The last Certificate in the chain should be issued by some authority "known" to client software - usually, some Root Authority.

If your CA Chain contains several separate PEM-encoded Certificates, enter all of them into the Certificate Authority Chain field. The Certificate issued by a Root Authority should be the last one in the list.

Click the Set CA Chain button to assign the Certificate Authority Chain to the Domain. If all Certificates in the Chain are decoded successfully and their format is correct, the CA Chain list is displayed:

| Certificate Authority Chain (Optional) | | | | |
|---|---|---|---|---|
| **Issued to** | | **Issued by** | **From** | **Till** |
| Organization VeriSign Trust Network<br>Unit      VeriSign, Inc.<br>Unit      VeriSign International Server CA - Class 3<br>Unit      www.verisign.com/CPS Incorp.by Ref. LIABILITY<br>       LTD.(c)97 VeriSign | | Country     US<br>Organization VeriSign, Inc.<br>Unit      Class 3 Public Primary<br>       Certification Authority | 17-Apr-<br>97 | 08-Jan-<br>04 |

**Note:** CommuniGate Pro checks only the format of each Certificate in the Chain. It does not check that each Certificate really certifies the issuer of the previous Certificate and that the last certificate in the Chain is issued by a Root Authority.

When set, the Certificate Authority Chain is sent to clients together with the Domain Certificate.

Click the Remove CA Chain button to remove the Certificate Authority Chain from the Domain Security Settings.

# Using Self-Signed Certificates

You can create a Self-Signed Certificate if you do not want to use any external Certificate Authority.
Click the Generate Self-Signed button and the CommuniGate Pro Server creates a Self-Signed certificate for you: the Issuer will be same entity you have specified, and the entire Certificate will be signed using the Domain Private Key. When a Domain has a Self-Signed Certificate, client applications will warn users that the addressed server has presented a certificate "issued by an unknown authority". Users can "install" self-signed certificates to avoid these warnings.

When client applications receive a Certificate and its issuer is not included into their list of Trusted Authorities, the applications may display warnings or they may refuse to accept the Certificate.

Your users can "install" your Domain Certificates into their Trusted Authorities lists. Once installed, the Certificate becomes a "trusted" one. For some programs (such as Mac versions of Microsoft Outlook and Outlook Express) installing an "untrusted" Certificate is the only way to use that Certificate for secure communications.

To install a Domain Certificate, the user should use a browser application and open the login page of the [WebUser Interface](#) for the selected Domain. If the Domain has an enabled Certificate, the Secure Certificate link appears. The user should click on that link to download the Domain Certificate and "open" it. The browser should allow the user to verify the Certificate and to install it into the list of Trusted Authorities.

When a Domain has a Self-Signed Certificate, the Refresh Self-Signed button appears on the WebAdmin page. Click this button to create a new Self-Signed Certificate with the same serial number, but with a new validity period.

# Trusted Root Certificates

The CommuniGate Pro Server can verify validity of Certificates presented to it. For example, the [WebUser Interface](#) performs validity checks when displaying signed messages.

A Certificate is considered valid if:

- it is equal to one of the Trusted Certificates, or
- it is issued by a holder of one of the Trusted Certificates.

There are several sets of the Trusted Certificates:

- Built-in trusted Certificates: these certificates are installed with the CommuniGate Pro Server software, and they are updated when the Server software is updated.
- Server-wide and Cluster-wider Trusted Certificates.
- Domain-wide Trusted Certificates.

When a PKI operation is performed for a certain Domain (or for a certain Account in that Domain), the following Trusted Certificates are checked:

- the Domain Trusted Certificates
- the Cluster-wide Trusted Certificates if the Domain is a Shared one, and the Server-wide Trusted Certificates if the Domain is not a Shared one
- the built-in Trusted Certificates

When a PKI operation is performed for the System itself (for example, an outgoing TLS connection is being established), the following Trusted Certificates are checked:

- the Server-wide Trusted Certificates
- the Cluster-wide Trusted Certificates
- the built-in Trusted Certificates

Use the WebAdmin Interface to update the set of Server-wide and Cluster-wide Trusted Certificates. Open Security page in the the Users realm. The Trusted Certificates page will open:

| Issued to | Serial Number | From | Till |
|---|---|---|---|
| RSA Data Security, Inc. | 02AD667E4E45FE5E576F3C98195EDDC0 | 09-Nov-94 | 08-Jan-10 |
| Thawte Personal Freemail CA | 00 | 01-Jan-96 | 01-Jan-21 |
| Thawte Personal Basic CA | 00 | 01-Jan-96 | 01-Jan-21 |
| My Company CA | 010256FF | 01-Jan-96 | 01-Jan-21 |

Trusted Certificates included into the displayed set have a checkbox marker. To remove certain Trusted Certificates, select its checkbox and click the Remove Marked button.

In addition to the certificates from the displayed set, the Domain-wide pages display the built-in Trusted Certificates, and the Trusted Certificates from the Server-wide set (or from the Cluster-wide set for Shared Domains).
The Server-wide and Cluster-wide Trusted Certificates pages display the the built-in Trusted Certificates.
These additional certificates do not have checkbox markers.

| Enter a PEM-encoded Certificate |
|---|
|  |

To add a Certificate to the set, enter the PEM-encoded Certificate data into the text field and click the Set Certificate button. The new Certificate should appear in the displayed set.

---

# SSL/TLS Secure Connections

The TLS (Transport Level Security) protocol a PKI application used to provide security and integrity of data transferred between a pair of communicating parties. The parties use PKI encryption to securely exchange a "secret key" data, and then all data transferred between the parties is encrypted using that "secret key". The earlier versions of the TLS protocol were called the SSL (Secure Socket Layer) protocols.

The CommuniGate Pro Server supports SSL/TLS connections for all its TCP-based services and modules. Secure connections can be established in two ways:

- A client application uses a special port to connect to the Server and starts to establish a TLS (encrypted) connection right after a TCP connection with that port is established. This method is used in all Web browsers when a `https://` URL is used.
  This method is also used when SIP clients use the "TLS transport" option.
  Some mail clients use it for POP, IMAP, LDAP, and (rarely) for SMTP connections.
  To support these clients, configure the Listeners for HTTP, SIP, POP, IMAP, SMTP, and LDAP modules: the Listeners should accept TCP connections on those special ports (see the module descriptions for the proper Secure Port Numbers) and the Init SSL/TLS option should be enabled for those ports.
- A client application uses a standard port to connect to the Server, and then it issues a special command (usually called STARTTLS or STLS) over the established clear-text TCP connection. When the Server receives such a command it starts to establish a secure connection. To support these clients you do not have to configure additional ports with the module Listeners.

Usually certificates for SSL/TLS communications can be assigned only to the CommuniGate Pro Domains that have at least one assigned network (IP) address. This limitation comes from the design of the TLS protocols used today: when a client application wants to initiate a secure connection, the Server has no information about the Domain the client wants to connect to. The Server knows only to which local IP address the client has connected, so it opens the Domain this IP address is assigned to, and uses the PKI Settings of that Domain.

An exception to this rule is the XMPP protocol. Before an XMPP client sends the `<starttls>` command, it explicitly specifies the target domain in the `<stream>` data, so the Server can initiate a TLS session with a Domain that has no assigned network address.

Use the WebAdmin Interface to specify the Server-wide SSL/TLS processing parameters. Open the General pages in the Settings realm, and find the TLS Sessions panel on the Others page:

| TLS Sessions | | |
|---|---|---|
| Log Level: Major & Failures | CBC Ciphers for old TLS | Process Target Domain extensions |
| Time to Live: 30 seconds | Weak Ciphers | Accept SSLv2 'hello' |
| Oldest Accepted: TLSv1.0 | | Abort on Wrong Client Certificate |

Log

Use this setting to specify what kind of information the TLS module should put in the Server Log. The TLS module records in the System Log are marked with the TLS tag.

Time To Live
> This setting specifies the *cache time* for TLS sessions. When all connections using the same TLS session are closed, the Server waits for the specified time before deleting the TLS session parameters. This feature allows clients to open new connections *resuming* the old TLS sessions. It increases connection speeds and decreases the Server CPU load. This feature is especially important for HTTP clients that open and close connections very often.

Oldest Accepted
> This setting specifies the oldest SSL/TLS protocol version accepted. If a remote peer specifies that it only supports an SSL/TLS version older than this one, an attempt to create a secure connection is rejected.
> This option controls incoming TCP connections only.

Process Target Domain extensions
> When this setting is enabled, the Server supports the TLS protocol extensions which allow the client to specify the name of the Server Domain it wants to access. This feature allows you to host several TLS-enabled Domains using the same Server Network IP Address.

CBC Ciphers for old TLS
> Select this setting if you want to support CBC-based cipher methods for SSL 3.0 and TLS 1.0 protocols. The CBC-based cipher methods are always supported for datagram (DTLS) protocols.

Weak Ciphers
> Select this setting if you want to support weak (less than 128-bit) security (cipher methods). The CBC Ciphers setting should be selected, too.

Accept SSLv2 'hello'
> If this setting is enabled, the Server accepts SSLv2-style initial connection requests used by many older clients.
> **Note:** even if this option is enabled, the actual security protocol used is SSLv3 or TLS, the SSLv2 security protocol is always disabled. There is no real reason to disable this option (other than to pass some bogus "security checks").

Abort on Wrong Client Certificate
> When a Server requires a client to send a Certificate, the client may send an incorrect certificate: expired, issued by a wrong issuer, etc.
> If this option is enabled, this situation aborts the current TLS negotiation process. If this option is disabled, then incorrect certificates are ignored.

# Client Certificates

The CommuniGate Pro Server can request a Client Certificate when an external client (a mailer, browser, or a real-time device) establishes a TLS connection with a certain Domain.

Use the WebAdmin Interface to open the Domain Settings page for the target Domain, and click the Security link. The PKI page will appear:

| Request Client Certificates | | |
|---|---|---|
| Issued by: Certificate Name 1 | Required: | default(No) |

Issued By
> Select one of the Trusted Certificates specified for this Domain.
> When a Trusted Certificate is selected, all TLS clients connecting to this Domain will be requested to present a valid certificate issued by the owner of the selected Trusted Certificate. These certificates can be used for Certificate Authentication.

Required
> If this option is selected, only clients presenting valid Certificates can establish TLS connections to this Domain.

The CommuniGate Pro Server processes Client Certificate requests when it establishes a TLS connection to remote servers (SMTP, XMPP, SIP, POP, and other connections). It processes these requests by sending the TLS Certificate assigned to the Main Domain.

# S/MIME Functionality

S/MIME is a PKI application used to digitally sign and encrypt E-mail and other messages. While TLS ensures data security when information is sent over an unprotected network, such as the Internet, S/MIME provides end-to-end data security: an S/MIME message is encrypted by the sender (using Multiparty Encryption) and submitted to the sender's server in the encrypted form. The same encrypted form is used when the message is transferred over a network, when it is stored on an intermediate server, and when it is deposited in the recipients' Mailboxes. Only the recipients can decrypt the message using their Private Keys, and only when they actually read the message: the message stays encrypted in the recipient's Mailboxes.

To use end-to-end S/MIME security, individual users should have their own PKI keys. Each user should have a Private Key securely stored in a storage available to that user only, and a matching Public Key embedded into a Certificate. This Certificate should be issued by a Certificate Authority that other users trust.

CommuniGate Pro WebUser and XIMSS Interfaces support S/MIME functions. The Server provides secure storage for user Private Keys. These Keys can be unlocked and used only by the users themselves, using these Interfaces.

To use a traditional desktop client application (a POP, IMAP, or MAPI client) the user Private Key should be stored in the PKI storage of the desktop operating system.
The WebUser and XIMSS Interfaces can export and import Private Keys, so the user can use the same Private Key for desktop applications and when employing these Interfaces. See the Secure Mail section for more details.

# Domain S/MIME Settings

The CommuniGate Pro Server implements a Certificate Server, issuing Certificates for its users.

A CommuniGate Pro Domain can act as a Certificate Authority for all its Accounts if:

- The Domain PKI Functions are Enabled.
- The Domain has a valid Private Key.
- The Domain has a valid generic Certificate or a special S/MIME Certificate.

To specify Domain S/MIME settings, use the WebAdmin Interface to open the Domain Settings pages. Open the Security page and click the S/MIME link. If the Domain has a valid Private Key, a page similar to those for the generic Domain Certificate are displayed. These fields can be used to enter a special S/MIME certificate for the Domain. This Certificate is used as the Issuer (Certificate Authority) for all S/MIME Certificates requested by users in this Domain.

If the special S/MIME Certificate is not specified, then the generic Domain Certificate is used as the Issuer Certificate.

# Automatic S/MIME Encryption

The S/MIME features can be used to provide secure message store. CommuniGate Pro can encrypt all or certain messages before it stores them in user Mailboxes.

The `Store Encrypted` Rule action is used to encrypt incoming E-mail messages and store them in the specified Mailbox.

Messages are encrypted using the S/MIME Certificate of the Mailbox owner. If the `Store Encrypted` Rule action is used in an Account-Level (i.e. in an Account or in a Domain-wide) Rule, and the specified Mailbox does not belong to the user Account, the message is encrypted using the Certificates of the Mailbox owner and the current Account.

Sample:
The Account john has a Rule with the following action:
`Store Encrypted in    ~jim/INBOX`
When this action is taken, the message is stored in the INBOX Mailbox of the Account jim encrypted with both john and jim Certificates. Both jim and john can decrypt and read this message.

# Stored Message Encryption

After CommuniGate Pro users receive certain clear-text E-mail messages, they may prefer to keep them encrypted in the Server Mailboxes. The MAPI and XIMSS clients, and the WebUser Interface provide the Encrypt and Decrypt functions that allow users to

encrypt and decrypt individual messages in their Mailboxes.

# DKIM Message Signing

DKIM (stands for DomainKeys Identified Mail) is an E-mail authentication method which allows for a unique Signature to be added for each E-mail message you send. The Signature is generated by using encryption with public and private keys. The receiving mail server can use the public key to determine if your private key was used to generate the message's Signature, and that the message body and the most important headers were not altered during the transit.

To specify Domain DKIM settings, use the WebAdmin Interface to open the Domain Settings pages. Open the Security page and click the DKIM link.

To enable adding the Signatures to outgoing messages you need to complete the following steps:

1. Generate (or import externally-generated) Private Key in the same way as assigning a Private Key for the Domain. After the Private Key is assigned the corresponding Public Key will be displayed.

| Private Key |
|---|
| Key Size:  1024 bit |
| Encryption Test: *Verification String is OK* |
| **Public Key** |
| |

2. Specify the value for `Domain` - it should be either the curent Domain name, or its "organizational domain" name.
3. Choose and secify the valuse for `Selector` - an arbitrary string.

| DKIM Signature Parameters | |
|---|---|
| Domain: | `mail.dept1.company.com` |
| Selector: | `mail` |

4. Create DNS TXT record for name *Selector*.`_domainkey`.*Domain* with value "v=DKIM1; p=MIGfMA0GCSqG..." where the value of "p=" is your Public Key.
5. Use `Check` button to verify that the DNS record exists and contains the right Public Key. Remember that newly created DNS records may need time to propagate through the world.

| DNS Record |
|---|
| |

6. After all the above steps are complete switch the `Enabled` to `Yes`

| Sign Outgoing Messages |
|---|
| Enabled:   `Yes` |

It is recommended to rotate the Private Key regularly (about every 3 months), by generating new Keys and creating new DNS record for a new `Selector`.

**Note:** The DKIM Signatures are composed by Enqueuer component when the message is enqueued, but physically added when the message is sent, stored or sent to an external program.

**Note:** For the DKIM signing process the relation of a message to a Domain is determined by the message's `From:` address. A user whose Account is in some other Domain may change his `From:` address so his messages will be signed according to the current Domain settings. Also, messages being relayed from external sources may be signed by the current Domain if their `From:` addresses match the names of the objects in the current Domain.

**CommuniGate Pro**

| Sys Admin | Logs | Router | Protection | Security | PKI | Intercept | Scalability | Alerts | Statistics |

Introduction
Installation
SysAdmin
Network
Objects
Storage
E-mail
Real-Time
Access
Services
Directory
Clusters
Applications
Miscellaneous
Licensing

WebMail
Pronto!
PBX

# Lawful Interception

- **Configuring Interception Settings**
- **Report Message Formats**

The CommuniGate Pro Server implements *Lawful Interception* - the functionality that plays a crucial role in helping law enforcement agencies to combat criminal activity.

The Server Administrator can specify the names of CommuniGate Pro Accounts that should be monitored. All login operations with those Accounts, all message manipulation activity, and all Real-Time activity in those Accounts is reported.

Reports can be sent as E-mail messages sent to specified addresses.

The reports can be sent to external programs via the PIPE module. Those programs can convert the reports generated with the CommuniGate Pro Server into the format required by the local law enforcement agencies.

Alternatively, reports can be sent using the PacketCable 2.0 protocol. Only the Real-Time and Media activity can be reported using that protocol.

# Configuring Interception Settings

To configure the Interception settings, open the Intercept page in the Master realm of the WebAdmin Interface:

| Account Name | Send Reports to | Court Case ID | Login | Signals | Media | New Mail | Sent Mail | Mailbox | Access | Partial | Deleted Mail |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | |

To add an element to list, fill the Account Name field in the last empty row and specify:

- an E-mail address to send the reports to, or
- the `pkcable:` prefix, the IP address and port of PacketCable collecting server, and, optionally, the slash (`/`) symbol and the IP Address and port of the PacketCable 2.0 media-collecting server.

Then specified the Court Case ID ordering the Lawful Intercept operations, select the checkboxes for the reports your need to generate, and click the Update button.

To remove an element, enter an empty line into the Account Name field and click the Update button.

In a Dynamic Cluster environment, the links for Server-wide and Cluster-wide pages appear. Enter the account names on the Cluster-wide page if accounts belong to Shared Domains served by the entire Cluster.

**Note:** If an element with an Account name has been added, removed, or modified, and the specified Account is currently in use, the changes will take effect on that Account within 1 minute.

If the Account name specified in an element is renamed, the element is automatically updated with the new Account name. If the Account name specified in an element is removed, the element is automatically removed.

# Report Message Formats

When generated reports are sent via E-mail, the report messages are composed using the following formats.

Login Report

The Login report is sent when a monitored user logs into the System. The report message has the `text/plain` format and contains the information about:

- The Account (User) name.
- The network (IP) address and the port the user logged in from
- The Protocol used (POP, IMAP, WebUser, XIMSS, SIP, XMPP, etc.)

Signal Report

The Signal report is sent when a monitored user sends or receives a Signal request or response. The report message has the `text/plain` format and contains the information about:

- The Account (User) name.
- The object type (Request or Response).
- The network (IP) address the object is received from.
- The Request or Response data.

New Message Report

The New message report is sent when a message is added to any Mailbox in the monitored Account.

The report message format is `multipart/mixed`. The first part has the `text/plain` format and contains the information about:

- The Account (User) name.
- The Action name (`attached message stored`)
- The Protocol, network (IP) address, and the port the user logged in from
- The Mailbox name
- The UID assigned to this message
- The time stamp
- The source of the message (incoming mail, copied from other Mailbox, appended, etc.)

The second part is a `message/rfc822` part and it contains a copy of the message added to the Mailbox.

Mailbox Report

The Mailbox report is sent when a monitored user creates, renames, or removes a Mailbox. The report message has the `text/plain` format and contains the information about:

- The Account (User) name
- The Action (Mailbox created, Mailbox renamed, Mailbox removed)
- The Protocol, network (IP) address, and the port the user logged in from
- The Mailbox name and, for the rename operation, the new Mailbox name

Access Report, Partial Access

The Access report is sent when a monitored user reads a message from one of the Account Mailboxes. The Partial Access report is sent when a monitored user reads a portion of a message (the message header, a message subpart, etc.) The report message has the `text/plain` format and contains the information about:

- The Account (User) name
- The Action (message read, messages deleted)
- The Protocol, network (IP) address, and the port the user logged in from
- The Mailbox name
- The UID(s) of the message(s) read or deleted. For partial access reports, the message portion specification is included, too

Sent Message Report

The Sent Message report is sent when a monitored user submits a new message.

The report message format is `multipart/mixed`. The first part has the `text/plain` format and contains the information about:

- The Account (User) name
- The Action (message sent)
- The Protocol, network (IP) address, and the port the user logged in from
- The name of an authenticated user (if any); `(self)` means the monitored user name

The second part is an `application/x-envelope` part and contains a copy of the message envelope data.

The third part is a `message/rfc822` part and contains a copy of the submitted message.

Deleted Message Report

The Deleted Message report is sent when a monitored user removes a message from some Mailbox.

The report message format is `multipart/mixed`. The first part has the `text/plain` format and contains the information about:

- The Account (User) name
- The Action (attached message deleted)
- The Protocol, network (IP) address, and the port the user logged in from
- The Mailbox name
- The UID of the deleted message.
- The time stamp of the deleted message.

The second part is a `message/rfc822` part and contains a copy of the submitted message.

Introduction

Installation

SysAdmin

Network

Objects

Storage

E-mail

Real-Time

Access

Services

Directory

Clusters

Applications

Miscellaneous

Licensing

WebMail

Pronto!

PBX

Sys Admin | Logs | Router | Protection | Security | PKI | Intercept | **Scalability** | Alerts | Statistics

# Scalability

- **Serving Large Domains**
- **Handling High-Volume Local Delivery**
- **Supporting Many Concurrent Clients**
- **System Tuning**
  - **Setting the TCP TIME_WAIT time**
- **Handling High-Volume SMTP Delivery**
- **Estimating Resource Usage**
- **OS Limitations and OS Tuning**
  - **Solaris**
  - **Windows**
  - **Linux**
  - **FreeBSD**
  - **HP-UX**
  - **Mac OS X**

This section explains how CommuniGate Pro and the Server OS can be optimized for maximizing per-server capacity and performance.

For horizontal scaling and multi-server redundancy, the Cluster configurations should also be used.

# Serving Large Domains

If some Domains you serve have a large number of Accounts (5,000 or more), you should consider storing accounts in Account Subdirectories rather than in a flat domain directory. This recommendation is based on the method that file systems use to maintain the list of entries within a directory index, and the maximum recommended number of entries is largely dependent on the type of file system in use.

For example, a file system with a hashed directory index is capable of efficiently accessing more directory entries than those file systems that use only a flat file for directory indexing. Some file systems can easily access an index of over 50,000 entries, while others become sluggish at only 1,000.

This same principle also applies to sites with over 2,000 or more domains on the server or cluster. In this scenario, it is recommended to use Domain Subdirectories

You can store subdirectories on multiple logical volumes, if necessary for storage volume or performance - just replace the moved subdirectories with their symbolic links. You can also move domain directories from the Domains directory and replace them with symbolic links.

# Handling High-Volume Local Delivery

When the number of messages to be delivered to local CommuniGate Pro accounts is expected to be higher than 1 message/second, you should allocate more "processors" in the Local Delivery Module. This is especially important for environments that process heavy inbound SMTP traffic (often used as a performance test environment). Insufficient number of Local Delivery module processors (threads) may result in excessive Queue growth and large latency in message delivery. You should watch the Local Delivery module Monitor and allocate more processors (threads) to that module if you see that the module Queue size grows to more than 200-300

messages. Do not allocate additional threads if, for example, you have 10 Local Delivery processors and see the waiting Local Delivery queue of 200 messages: this Queue size introduces only 1-2 seconds delivery latency. Increase the number of Local Delivery threads only if you see that Queue growing.

Some types of storage arrays benefit from a significant number of parallel delivery threads. For example, there are some NFS arrays which can deliver messages more efficiently with 100 Local Delivery processors than with 10, given the same number of messages. The storage vendor should be requested to provide information about the optimal number of parallel write operations for each system accessing the array, and the number of CommuniGate Pro Local Delivery processors can be adjusted to hit this target number. As Local Delivery processors are static (the configured number of processors remain in existence consistently throughout the life of the process), it is important to configure enough processors but wasteful of system resources to configure vastly too many.

Administrators of heavily loaded servers may want to disable the Use Conservative Info Updates option (located in the Local Account Manager panel on the Others page in the WebAdmin Settings realm). Disabling this option decreases the load on the file I/O subsystem.

---

# Supporting Many Concurrent Clients

For larger installations, the number of users that can be served simultaneously is an issue of a very high concern. In order to estimate how many users you can serve at the same time, you should realize what type of service your clients will use.

POP3 Clients

> POP3 mailers connect to the server just to download new messages. Based on the average connections speeds, expected mail traffic, and your user habits, you can estimate how much time an average session would take. For example, if you are an ISP and you estimate that an average your "check mail" operation will take 15 seconds, and they mostly check their accounts an average of 2 times each during 12 peak hours, then with 100,000 POP3 users you can expect to see 100,000 * 2 * 15 sec / (12*60*60 sec) = ~70 concurrent POP3 sessions.

> This number is not high, but POP3 sessions put a high load on your disk I/O and network I/O subsystems: after authentication, a POP3 session is, essentially, a "file downloading" type of activity.

IMAP4 Clients

> The IMAP protocol allows a much more sophisticated processing than POP3. Mail is usually left on the server, and some unwanted messages can be deleted by users without downloading them first.

> The IMAP protocol is "mail access", not "mail downloading" protocol. IMAP users spend much more time being connected to the server. In corporate environments, users can leave their IMAP sessions open for hours, if not days. While such inactive sessions do not put any load on your disk or network I/O subsystems or CPU, each session still requires an open network connection and a processing thread in the server. Since the IMAP protocol allows users to request search operations on the server, IMAP users can also consume a lot of CPU resources if they use this feature a lot.

> When the server needs to handle many IMAP or POP connections, it is important to configure more IMAP and POP channels, to allow for large numbers of users to connect concurrently. Some modern IMAP clients and the MAPI connector may even open multiple connections for a single account, and each is counted in the IMAP channel total. Fortunately, IMAP and POP channels are created only when used, so no resources are consumed if the IMAP and POP channels are set to 10000 if only 2000 are being used - however, be careful to set this value below the threshold where your system will be unable to withstand further connections, and could become unresponsive for users already connected. The IMAP and POP channels setting provides a limit for protecting your system or cluster resources from being overwhelmed, in the case of a peak load or denial of service (DoS) attack.

WebUser Clients

> The CommuniGate Pro WebUser Interface provides the same features provided by IMAP mailer clients, but it does not require an open network connection (and processing thread) for each user session. When a client (a browser) sends a request, a network connection is established, the request is processed with a server thread, and the connection is closed.

> This allows the Server to use just 100 HTTP connections to serve 3,000 or more open sessions.

> CommuniGate Pro also supports the HTTP 1.1 "Keep-Alive" option, located on the WebUser Interface Settings page as "Support Keep-Alive". HTTP Keep-Alive sessions for WebUsers will cause each WebUser session to maintain one or more

open connections from the user client to the server for the entire session duration. This method is not recommended on a busy, high-volume server as it will consume significant CPU and operating system resources, but can be used to optimize WebUser response time for end users if the system can handle the additional overhead. Keep-Alive connections will only be offered on Frontend servers in a Cluster.

XIMSS Clients (including Pronto)

XIMSS clients may work directly, via a TCP connection - then the same considerations as those for IMAP clients should be applied. If XIMSS client work using HTTP Binding (so-callend "Proxy-safe mode"), then the limit for HTTP User connections should be increased. Most XIMSS clients keep one HTTP connection open all the time, in order to receive asynchronous messages from the server.

SIP/RTP Clients

As compared to messaging, which tends to be very disk I/O-limited, SIP and RTP comunications have real-time requirements and only a few actions (such as a SIP REGISTER) cause some disk I/O operations. Real-time traffic is highly susceptible to any network or system latency, and as such is more closely tied to CPU performance than E-mail transfer. However, these real-time requirements can be satisfied through today's ever increasing CPU and bus speeds.

In order to optimize SIP and RTP performance, your CommuniGate Pro Server should run on modern systems with adequate CPU and memory headroom. If most of the traffic through CommuniGate Pro is just SIP signaling traffic, then even a single-CPU server should be capable of upwards of 100 calls per second. However, when performing significant amounts of SIP and RTP proxying, NAT traversal, PBX functions and media termination, the demands on memory, network, and especially CPU will be significant.

Increasing the number of SIP Server and SIP Client processors, as well as Signal processors, is required.

These threads are all "static", meaning that the threads are created regardless of whether or not they are in use, and they will consume some memory resources for their stacks.

# System Tuning

When optimizing a system for performance, there are often certain system kernel and TCP/UDP stack tuning options available which allow the system to open more concurrent network connections and allow the CommuniGate Pro process to open many file descriptors. While most operating systems allow for tuning these options, the method of doing so will differ greatly across platforms, and you may need to contact your operating system vendor or research the proper way to modify your system accordingly.

The number of available file descriptors should be set to approximately 2x the number of concurrent IMAP, POP, SMTP, SIP/RTP, and other types of connections required. You should also be certain that the operating system is capable of efficiently opening this many TCP/UDP sockets simultaneously - some OSes have a "hash table" for managing sockets, and this table should be set greater than the number of sockets required. Often times, allowing at least 8192 sockets and 16384 open file descriptors per process should be plenty for most systems, even under significant load. Increasing these values much too high can in most cases consume some memory resources, and should be avoided. Setting the limit on the number of open file descriptors to "unlimited" in the shell can also cause problems on some operating systems, as this could set the available file descriptors to the 32-bit or 64-bit limits, which can in some cases waste memory and CPU resources.

## Setting the TCP `TIME_WAIT` time

When you expect to serve many TCP/IP connections, it is important to check the time your Server OS waits before releasing a logically closed TCP/IP socket. If this time is too long, those "died" sockets can consume all OS TCP/IP resources, and all new connections will be rejected on the OS level, so the CommuniGate Pro Server will not be able to warn you.

This problem can be seen even on the sites that have just few hundred accounts. This indicates that some of the clients have configured their mailers to check the server too often. If client mailers connect to the server every minute, and the OS TIME_WAIT time is set to 2 minutes, the number of "died" sockets will grow, and eventually, they will consume all OS TCP/IP resources.

While the default TIME_WAIT setting on many systems is often 120 or 240 seconds, some operating systems have begun setting a default TIME_WAIT value of 60 seconds, and it is probably safe to set `TIME_WAIT` time as low as 20-30 seconds.

The TIME_WAIT problem is a very common one for Windows systems. Unlike most Unix systems, Windows NT does not have a generic setting for the TIME_WAIT interval modification. To modify this setting, you should create an entry in the Windows NT

Registry (the information below is taken from the http://www.microsoft.com site):

- Run Registry Editor (RegEdit.exe)
- Go to the following key in the registry:
  `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tcpip\Parameters`
- Choose `Add Value` from the `Edit` menu and create the following entry:

  Value Name:
  > `TcpTimedWaitDelay`

  Data Type:
  > `REG_DWORD`

  Value:
  > 30-300 (decimal) - time in seconds
  > Default: 0xF0 (240 decimal) not in registry by default

- Quit the Registry Editor
- Restart the computer for the registry change to take effect.

Description: This parameter determines the length of time that a connection will stay in the TIME_WAIT state when being closed. While a connection is in the TIME_WAIT state, the socket pair cannot be reused. This is also known as the "2MSL" state, as by RFC the value should be twice the maximum segment lifetime on the network. See RFC793 for further details on MSL.

# Handling High-Volume SMTP Delivery

To handle high-volume (more than 50 messages/second) SMTP delivery load you need to ensure that your DNS server(s) can handle the load CommuniGate Pro generates and that the UDP packet exchange between CommuniGate Pro and the DNS servers does not suffer from excessive packet loss. You may want to re-configure your Routers to give UDP traffic a higher priority over the TCP traffic.

Use the WebAdmin Interface to fine-tune the DNS Resolvers settings. Open the Network pages in the Settings realm, then open the DNS Resolver page.

You may want to try various values for the Concurrent Requests: depending on your DNS server(s) setup, increasing the number of Concurrent Requests over 10-20 can result in DNS server performance degradation.

If an average size of the messages sent via SMTP is higher than 20K, you should carefully select the number of SMTP sending channels (threads), too. Too many concurrent data transfers can exceed the available network bandwidth and result in performance degradation. 500 channels sending data to remote sites with a relatively slow 512Kbit/sec connectivity can generate 250Mbit/sec outgoing traffic from your site. Usually the traffic is much lighter, since outgoing channels spend a lot of time negotiating parameters and exchanging envelope information. But as the average message size grows channels spend more time sending actual message data and the TCP traffic generated by each channel increases.

If using SMTP External Message Filters (Plugins) - such as anti-virus, anti-spam, or other content-filtering helpers - then the SMTP load can be optimized by putting any temporary file directories used by these plugins onto a memory or tmpfs filesystem, if your system has the available memory. Since all messages should be queued in the real CommuniGate Pro Queue directories, there should be no risk in putting the plugin temporary file directories, as long as those directories never contain the only copy of any message. Even in the event of an error, power failure, or server crash, any undelivered message should always be queued to "stable storage" in the normal CommuniGate Pro Queue directory.

# Estimating Resource Usage

Each network connection requires one network socket descriptor in the server process. On Unix systems, the total number of sockets and files opened within a server process is limited.

When the CommuniGate Pro server starts, it tries to put this limit as high as possible, and then it decreases it a bit, if it sees that the limit set can be equal to the system-wide limit (if the CommuniGate Pro consumes all the "descriptors" available on the server OS, this will most likely result in the OS crash). The resulting limit is recorded in the CommuniGate Pro Log.

To increase the maximum number of file and socket descriptors the CommuniGate Pro Server process can open, see the instructions

below.

Each network connection is processed by a server *thread*. Each thread has its own *stack*, and the CommuniGate Pro threads have 128Kbyte stacks on most platforms. Most of the stack memory is not used, so they do not require a lot of real memory, but they do add up, resulting in bigger *virtual memory* demand. Most OSes do not allow the process virtual memory to exceed a certain limit. Usually, that limit is set to the OS swap space plus the real memory size. So, on a system with just 127Mbytes of the swap space and 96Mbytes of real memory, the maximum virtual memory that can be allocated is 220Mbytes. Since the swap space is shared by all processes that run under the server OS, the effective virtual memory limit on such a system will be around 100-150MB - and, most likely, the CommuniGate Pro Server will be able to create 500-1000 processing threads.

On 32-bit computers, 4GB of virtual memory is the theoretical process memory size limit (and in reality this virtual memory limit for user-space processes is often only 2GB), and allocating more than 4GB of disk space for page swapping does not provide any benefit. Since memory has dropped in price significantly, 4GB of RAM memory is often recommended for 32-bit systems in order to maximize the available memory capacity, on those operating systems which allow a single process to utilize 2GB or more of virtual memory space. When supporting many concurrent IMAP, POP3, or SIP/RTP connections, the CGServer process will grow in size according to the per-thread stack space allocated, in addition to other memory needs. If supporting greater than 4000 processing threads, then an operating system should be considered which can allocate more than 2GB of virtual memory to the CGServer process, and 4GB of RAM memory should be installed on the system.

During a POP3 or IMAP4 access session one of the Account Mailboxes is open. If that Mailbox is a text file Mailbox, the Mailbox file is open. During an incoming SMTP session a temporary file is created for an incoming message, and it is kept open while the message is being received. So, on Unix systems, the total number of open POP, IMAP, and SMTP connections cannot exceed 1/2 of the maximum number of socket/file descriptors per process. For high-performing systems, you may want to consider allowing at least 8192 or more open file descriptors per process.

While a WebUser session does not require a network connection (and thus a dedicated socket and a thread), it can keep more than one Mailbox open. (If using HTTP Keep-Alive, then each WebUser session does consume at least one network connection, also.)

On Unix systems, when the Server detects that the number of open network sockets and file descriptors is coming close to the set limit, it starts to reject incoming connections, and reports about this problem via the Log.

---

# OS Limitations and OS Tuning

This section explains how to optimize the kernel and TCP tuning parameters available on some of the most common CommuniGate Pro platform Operating Systems.

The most commonly encountered limits are:

- Open file descriptors - the maximum number of files and network sockets a single process can open.
- The maximum size of virtual memory available to a process.

**Please always confirm these changes with your operating system vendor, and always test changes on a test system before using on a production server. Variations in operating system versions, patches, hardware, and load requirements can vary the best settings for these values. Examples are provided as a guide but may not always be optimal for every situation.**

## Solaris

Generally applicable to Solaris 8, 9, and 10

**CommuniGate Pro "Startup.sh" file**

Startup.sh is by default referenced at `/var/CommuniGate/Startup.sh`. You may need to create it. This file is read by the init startup script `/etc/init.d/STLKCGPro.init` to be executed at boot time.
The default Solaris `malloc` library is not very efficient in a multithreaded environment, especially when the Server has more than 2 CPUs, and the alternative `mtmalloc` library may provide better performance.
The following is a recommended Startup.sh file for larger Solaris implementations.

```
SUPPLPARAMS="--DefaultStackSize 131072 --closeStuckSockets --CreateTempFilesDirectly 10"
ulimit -n 32768
LD_PRELOAD=libmtmalloc.so
```

**ncsize**

Solaris `ncsize` kernel parameter has to be *decreased* on the large systems, especially - on Dynamic Cluster backends. The cache this parameter controls cannot keep any usable subset of file paths, but the large cache size causes the system to waste a lot of CPU cycles checking this cache table (symptoms: more than 50% CPU utilization, most CPU time is spent in the kernel). Decrease the `ncsize` kernel parameter value down to 1000-2000.

**Additions to `/etc/system`**

Following are a few settings appropriate for most Solaris systems, where significant load capacity is required. A good estimate is to set these values between 1-2 times the expected peak capacity.

```
* system file descriptor limit (setting the max setting to 32768 may
* be better in some instances)
set rlim_fd_cur=4096
set rlim_fd_max=16384
* tcp connection hash size
set tcp:tcp_conn_hash_size=16384
```

**Note:** `/etc/system` changes require a system reboot to take effect.

**Other kernel driver options:**

```
# solaris 9/10 uses a default of 49152
ndd -set /dev/tcp tcp_recv_hiwat 49152 # or 65536
ndd -set /dev/tcp tcp_xmit_hiwat 49152 # or 65536
# increase the connection queue
ndd -set /dev/tcp tcp_conn_req_max_q 512
ndd -set /dev/tcp tcp_conn_req_max_q0 5120
# decrease timers
ndd -set /dev/tcp tcp_fin_wait_2_flush_interval 135000
ndd -set /dev/tcp tcp_time_wait_interval 60000
ndd -set /dev/tcp tcp_keepalive_interval 30000
## naglim should likely only be disabled on Backends
## 1=disabled, default is 53 (difficult to confirm)
# ndd -set /dev/tcp tcp_naglim_def 1
```

# Windows 9x/NT/200x/XP/Vista

The Windows system limits the maximum port number assigned to outgoing connections. By default this value is 5000. You may want to increase that value to 20,000 or more, by adding the `MaxUserPort` DWORD-type value to the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters` key.

For more details, check the [Microsoft Support Article Q196271](#).

# Linux

**CommuniGate Pro "Startup.sh" file**

On Linux, the Startup.sh file is referenced by default at `/var/CommuniGate/Startup.sh`. You may need to create it. This file is read by the init startup script `/etc/init.d/CommuniGate` to be executed at boot time. The following is a Startup.sh file for CommuniGate Pro version 4.3 or later for Linux 2.4 or later. In some cases, you may find that more file descriptors are required, so the "ulimit -n" value can be increased up to 32768 safely, if necessary.

```
SUPPLPARAMS="--DefaultStackSize 131072 --useNonBlockingSockets --closeStuckSockets --CreateTempFilesDirectly
10"
ulimit -n 16384
```

**Linux kernel 2.6 and later:**

Linux kernel 2.6 introduced "POSIX threads", replacing the previous default thread library "LinuxThreads". The 2.6 kernel implementations are the first Linux release for which using POSIX threading is recommended. Following are some tuning options for Linux 2.6. For most Linux distributions, these shell commands should be placed into a boot script to be run at system startup. RedHat and a few other distributions may also provide a facility to configure "sysctl" data in the file `/etc/sysctl.conf`:

```
#!/bin/sh
# Linux 2.6 tuning script
# max open files
echo  131072 > /proc/sys/fs/file-max
# kernel threads
echo 131072 > /proc/sys/kernel/threads-max
# socket buffers
echo 65536 > /proc/sys/net/core/wmem_default
echo 1048576 > /proc/sys/net/core/wmem_max
echo 65536 > /proc/sys/net/core/rmem_default
echo 1048576 > /proc/sys/net/core/rmem_max
# netdev backlog
echo 4096 > /proc/sys/net/core/netdev_max_backlog
# socket buckets
echo 131072 > /proc/sys/net/ipv4/tcp_max_tw_buckets
# port range
echo '16384 65535' > /proc/sys/net/ipv4/ip_local_port_range
```

# FreeBSD

Following are some tuning optimizations applicable to different versions of FreeBSD.

**CommuniGate Pro "Startup.sh" file**

Startup.sh is by default referenced at `/var/CommuniGate/Startup.sh`. You may need to create it. This file is read by the init startup script `/usr/local/etc/rc.d/CommuniGate.sh` to be executed at boot time. The following is a Startup.sh file for CommuniGate Pro version 4.3 or later for most FreeBSD implementations. In some cases, you may find that more file descriptors are required, so the "ulimit -n" value can be increased up to 32768 safely, if necessary:

```
SUPPLPARAMS="--DefaultStackSize 131072 --useNonBlockingSockets --closeStuckSockets --CreateTempFilesDirectly
10"
ulimit -n 16384
```

A `/boot/loader.conf.local` file can be used to set boot-time kernel parameters:

```
# increase the TCP connection hash to a value just greater than peak needs
# (this can be set higher if necessary)
net.inet.tcp.tcbhashsize="16384"
```

The Loader configuration file `/boot/loader.conf` should be modified:

```
kern.maxdsiz="1G"                   # max data size
kern.dfldsiz="1G"                   # initial data size limit
kern.maxssiz="128M"                 # max stack size
kern.ipc.nmbclusters="65536"        # set the number of mbuf clusters
net.inet.tcp.tcbhashsize="16384" # size of the TCP control-block hashtable
```

**FreeBSD 5 and above**

Sysctl settings can be set automatically in the `/etc/sysctl.conf` file:

```
# cache dir locations, on by default
vfs.vmiodirenable=1
# increase socket buffers
kern.ipc.maxsockbuf=1048576
kern.ipc.somaxconn=4096
# increase default buffer size
net.inet.tcp.sendspace=65536
net.inet.tcp.recvspace=65536
# decrease time wait
net.inet.tcp.keepidle=300000
net.inet.tcp.keepintvl=30000
# increase vnodes
kern.maxvnodes=131072
# increase maxfiles/maxfiles per process
kern.maxfiles=131072
kern.maxfilesperproc=65536
# increase port range
net.inet.ip.portrange.last=65535
# default: net.inet.ip.rtexpire: 3600
net.inet.ip.rtexpire=300
# decrease MSL from 30000
net.inet.tcp.msl=10000
# increase max threads per process from 1500
kern.threads.max_threads_per_proc=5000
```

# HP-UX

HP-UX kernel parameters for HP-UX are set through a few different mechanisms, depending on the HP-UX version used. The following kernel parameters are important to increase higher than peak capacity needs:

```
maxfiles        Soft file limit per process
maxfiles_lim    Hard file limit per processes
maxdsiz         Maximum size of the data segment
nfile           Maximum number of open files
ninode          Maximum number of open inodes

# suggested parameter settings
maxfiles        4096
maxfiles_lim    32768
maxdsiz         (2048*1024*1024)
nfile           32768
ninode          32768
```

Decreasing the TCP TIME_WAIT parameter is also recommended:

```
ndd -set /dev/tcp tcp_time_wait_interval 60000
```

# Mac OS X

The Mac OS X sets a 6MB limit on "additional" virtual memory an application can allocate. This is not enough for sites with more than several thousand users, and you should increase that limit by specifying the following in the CommuniGate Pro Startup.sh file:

```
ulimit -d 1048576
ulimit -n 10000
```

Introduction

Installation

SysAdmin

Network

Objects

Storage

E-mail

Real-Time

Access

Services

Directory

Clusters

Applications

Miscellaneous

Licensing

WebMail

Pronto!

PBX

Sys
Admin | Logs | Router | Protection | Security | PKI | Intercept | Scalability | Alerts | Statistics

# Alerts

- **Posting Alerts**
- **Storage Quota Alerts**

The CommuniGate Pro Server can send Alert messages to its users.

Alerts are displayed to the users when they connect to POP module or when they work with their Accounts using the IMAP module, XIMSS module, or the WebUser Interface.

Alerts can be posted by the Server and/or Domain Administrator, and some alert messages can be generated automatically by the CommuniGate Pro Server software.

The Trigger Handlers can use Account Alerts to notify System Administrators about certain system events.

# Posting Alerts

Server and Domain administrators can also send Alert messages to all CommuniGate Pro Domain users, and to an individual CommuniGate Pro Account user. The Server and Cluster Administrators can also send Alert messages to all CommuniGate Pro users.

To send an Alert Message, the administrator should follow the Alerts link either on the Domains page (for Server-wide and Cluster-wide Alerts), or on the Domain Settings page (for Domain-wide Alerts), or on the Account Settings page (for alerts sent to an individual Account).

The Alerts page appears and lists the already posted Alerts:

| Posted Alerts | |
|---|---|
| **2-Aug-2007** | Server will be shut down on Aug,3 from 1:00pm till 1:30pm |
| 22:57:13 | Please check your mailer - we will enforce secure authentication starting Aug, 5th |
| **23:53:28** | The next service shut down is scheduled on Aug, 15th |
| 23:54:10 | The IMAP service is now available |

The Alerts page for a CommuniGate Pro Domain displays both Server-wide and Domain-wide alerts. The Server-wide Alerts have highlighted (bold) time stamps, and they cannot be removed using the Domain Alerts page.

To post an Alert message, enter the message text in the text field and click the Post Alert button.

To remove some Alert messages, mark them using the checkboxes and click the Remove Marked button.

A Domain administrator can add and remove Domain and Account Alerts only if the `CanPostAlerts` access right is granted to the administrator Account.

In a Dynamic Cluster the system maintains separate Server-wide and Cluster-wide Alert sets. The Server-wide Alerts are displayed to all users with the Accounts in non-Shared (Local) Domains, while the Cluster-wide Alerts are displayed to all Shared Domain Account users.

Alerts sent to an individual Account are removed as soon as they are delivered to the Account user. Old and outdated Domain-wide, Server-wide, and Cluster-wide Alerts should be explicitly removed by administrators.

# Storage Quota Alerts

The Server checks the Account storage quota for every connected user. If the Account storage is limited, and the specified percent of that limit is already used, the Server generates an alert message for that user.

The Account settings specify when the Storage Quota Alerts should be generated.

After a Storage Quota Alert is sent to the Account user, the Server does not generate Storage Quota Alerts for that Account for 10 minutes.

**Note:** if a user connects to his/her Account using the POP module, the Storage Quota Alert is displayed as an error message, and the user should try to connect again. If the user does not retry immediately, but makes the next connection attempt more than 10 minutes later, and the Account is still over its storage quota, the Storage Quota Alert is generated again and the connection is refused again. Instruct your POP3 users to retry immediately if they see the Storage Quota Alert messages.

| Introduction |
| --- |
| Installation |
| SysAdmin |
| Network |
| Objects |
| Storage |
| E-mail |
| Real-Time |
| Access |
| Services |
| Directory |
| Clusters |
| Applications |
| Miscellaneous |
| Licensing |
| |
| WebMail |
| Pronto! |
| PBX |

Sys
Admin Logs Router Protection Security PKI Intercept Scalability Alerts Statistics

# Statistics

- **Monitoring Statistics Elements via Web**
- **Monitoring Statistics Elements via CLI/API**
- **Trigger Manager**
  - **Configuring Trigger Handlers**
  - **Notification via E-mail**
  - **Notification via Instant Messages**
  - **Notification via SNMP Traps**
  - **Notification via Account Alerts**
  - **Notification via URL Requests**
  - **Frequency Limits**
  - **Specifying Triggers**
- **Logging**
- **Custom Statistics Elements**

The CommuniGate Pro Server maintains a set of Statistics Elements containing information about the Server activity. These Elements can be be accessed via the built-in SNMP server ("SNMP agent"), Network CLI/API, CG/PL applications, and the WebAdmin Monitor pages.

# Monitoring Statistics Elements via Web

Use the WebAdmin Interface to monitor the Server Statistics Elements via Web. Open the Statistics page in the Monitors realm.

| Filter: | | **MIB** | 30 of 248 selected |
| --- | --- | --- | --- |
| smtpInputActive | 0 | smtpInputTotal | 0 |
| smtpInputJobs | 0 | smtpInputTrafficIn | 0 |
| | .................. | | |

The page contains the list of the Statistics Elements and their current values. Each Element name is a link. Click the Element link to open the Element Monitor page.

You can use the Filter field and the Display button to filter Statistics elements by name.

The Element Monitor page contains a histogram allowing you to monitor how the Element value changes over time:



**Total:** 451K
**Current:** 152/sec
300
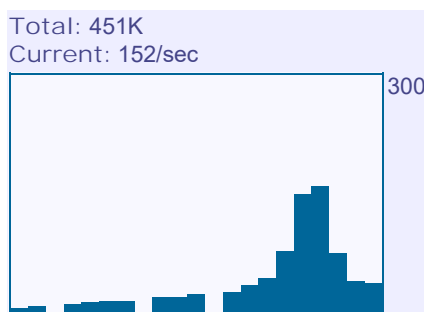
There are time stamps at the bottom of the histogram (the time stamp on the right is the latest sampling time). On the right side of the histogram the graphic scale is indicated.

The histogram traces the current value of the INTEGER-type Elements. For the COUNTER-type Elements, the histogram traces the difference between the last two sample values, divided by the number of seconds passed between samples.

The WebAdmin Preferences can be used to change the parameters of the Statistics Web Monitor subsystem.

# Monitoring Statistics Elements via CLI/API

The CLI/API GETSTATELEMENT command allows a Server Administrator to monitor the Statistics Elements via the CLI/API interface and various CLI "wrappers".

# Trigger Manager

The Trigger Manager monitors the values of selected Statistics Elements. A Trigger is "released" when the value of an INTEGER-type Element crosses the specified threshold, or when the value of a COUNTER-type Element has increased more than the specified limit over the specified period of time. For example, a Trigger can be "released" when there are more than 10,000 E-mail messages in the Server Queue (the value of the numQueuedMessages INTEGER-type Statistics Element is over 10,000) or when the SIP Module has to process more than 500 incoming requests per second (the value of the sipTotalServers COUNTER-type Statistics Element increased for more than 5000 during the last 10 seconds).

## Configuring Trigger Handlers

A Trigger Handler specifies how a System Administrator or System Operator should be notified. Several notification methods are supported, and each Trigger Handler can use any number of supported methods.

Use the WebAdmin Interface to configure the Trigger Handlers. Open the General pages in the Settings realm, then open the Triggers pages.
Click the Handler link to configure the Trigger Handlers. The list of existing Trigger Handlers will appear. Each Trigger Handler has a name and notification methods with parameters:

| Name: |
| :---: |
| .....methods.... |

There is an empty Trigger Handler at the bottom of the page. Enter a new Handler name and click the Update button to create a new Handler.

To remove a Handler, empty its Handler Name field and click the Update button.

To rename a Handler, change the value of its Handler Name field and click the Update button.

# Notification via E-mail

To send Trigger Notifications via E-mail, select the Send E-mail option:

| Send E-mail | Subject: |
| | To: |

`Subject`
> This field specifies the Subject of E-mail messages sent with this Handler.

`To`
> This field specifies the recipient(s) for E-mail messages sent with this Handler. Multiple recipients should be separated with the comma (`,`) symbol.

*Body*
> This field specifies the text of E-mail messages sent with this Handler.

The Subject and Body parameters can include the special symbol combinations:

`^0`
> This combination is replaced with the name of the Statistics Element that has released the Trigger.

`^1`
> This combination is replaced with the value of the Statistics Element threshold.

`^2`
> This combination is replaced with the actual value of the Statistics Element.

`^3`
> This combination is replaced with the current Server time.

## Notification via Instant Messages

To send Trigger Notifications as an Instant Message, select the Send Instant Message option:

| Send Instant Message | To: |
| --- | --- |
| | |

`To`
> This field specifies the recipient for Instant Messages sent with this Handler.

*Body*
> This field specifies the text of Instant Messages sent with this Handler. This text can contain the same special symbol combinations that can be used in E-mail Notifications (see above).

## Notification via SNMP Traps

To send Trigger Notifications as [SNMP Traps](#), select the Send SNMP Trap option:

| Send SNMP Trap | To: |
| --- | --- |
| | Active Monitors |

`To` *address field*
> This field specifies the addresses for computers to which SNMP Traps should be sent. Multiple addresses should be separated using the comma (`,`) symbol. Addresses can be specified as network (IP) addresses or as DNS domain names. If you want to send SNMP Traps not to the standard UDP port 162, but to a different port, specify the port number after the address, using the colon (`:`) symbol.

`To` *Active Monitors*
> If this option is selected, the SNMP Trap is sent to all "Active SNMP Monitor" computers - all computers that have recently sent

requests to the CommuniGate Pro SNMP module using the Trap Password "community name".

## Notification via Account Alerts

To send Trigger Notifications as Account Alerts, select the Send Account Alert option:

| Send Account Alert | To: |
|---|---|
|  |  |

To
> This field specifies the names of CommuniGate Pro Accounts the Alert should be sent to. Multiple names should be separated using the comma (,) symbol.

Text
> This field specifies the text of the Alert. This text can contain the same special symbol combinations that can be used in E-mail Notifications (see above).

## Notification via URL Requests

To send Trigger Notifications a URL requests, select the Send URL option:

| Send URL | To: |
|---|---|

To
> This field specifies the URL this Handler should send a request to. The text can contain the same special symbol combinations that can be used in E-mail Notifications (see above).

## Frequency Limits

Each Trigger Handler has settings limiting the amount of notifications the Handler can generate. This limit is necessary, because some Element can get into the "red zone" (cross the threshold) and stay there for some period of time. Without a limit, the Trigger Handler would send notifications every time it sees the Element in the "red zone" (approximately every 5 seconds).

| Frequency: | not more than | 2 | within | 15 minutes |
|---|---|---|---|---|

If the Handler has already sent the specified number of notification during the specified period of time, no more notification is sent, but the released Triggers are still recorded in the Server Log.

## Specifying Triggers

You can specify a Trigger by setting a "threshold" for some Statistics Element.

For an INTEGER-type Element you specify the threshold as an integer value: if the Element value becomes larger than the threshold value, the Trigger is released.

For a COUNTER-type Element you specify the threshold as an integer value and a time period: if the Element value has increased for more than the threshold value during the specified period of time, the Trigger is released.

Use the WebAdmin Interface to specify Triggers. Open the General pages in the Settings realm, then open the Triggers pages. Click the Elements link to specify Triggers:

| Element | Handler | Threshold |
|---|---|---|
|  | ........................... |  |
| smtpInputActive | Warning | 1000 |

| Element | | Handler | Threshold | | |
|---|---|---|---|---|---|
| smtpInputTotal | --- | | --- | | |
| smtpInputJobs | --- | | 10000 | in | minute |
| smtpInputMessagesReceived | --- | | --- | in | --- |
| | ........................... | | | | |

The Handler field specifies which Trigger Handler should be used to process this Trigger.

To remove a Trigger, reset its Threshold value (set it to ---).

If you want to disable a Trigger without removing its threshold value, reset its Handler field (set it to ---).

# Logging

You may want to record the value of all Statistics Elements periodically.

Open the General pages in the Settings realm, then open the Triggers pages. Click the Events link and scroll to the bottom of the page:

| Element | Handler | Threshold |
|---|---|---|
| ........................... | | |
| Log all Element values every: | 15 sec | |

Select a desired time period and click the Update button. A record will be created in the "Statistics" Supplementary Log.

Each record contains the values of all Statistics Elements separated with the tabulation symbols.

The very first record of each file starts with the star (*) symbol and contains the Statistics Element names, also separated with the tabulation symbols. This record is also inserted after the CommuniGate Pro Server restarts, as the new version may have additional Statistics Elements.

# Custom Statistics Elements

There are 10 custom COUNTER-type Elements named custom1, custom2, ... custom10. The Server components do not modify their values themselves, but they can be updated using the CG/PL functions.

CommuniGate® Pro Guide. Copyright © 1998-2018, Stalker Software, Inc.

| Network | Listeners | Dialup |
|---|---|---|

# Network

- LAN Addresses
- Port Allocation
- NATed Addresses
- NAT/Firewall Parameters
- Domain Name Resolver (DNR)
- IPv6 Support
- Network Address Lists
- Denied Addresses
- Debug Addresses

CommuniGate Pro is a network server, and it needs to know the configuration of your network. Most of the settings are retrieved automatically from your OS setup, but you may want to change these settings and/or specify additional settings.

This section describes the CommuniGate Pro network settings.

# Network Address Lists

Many CommuniGate Pro components use Network (IP) Address lists. These lists are specify Client and Blacklisted addresses, access restrictions for Listeners, etc.

This section describes the Network Address List format.
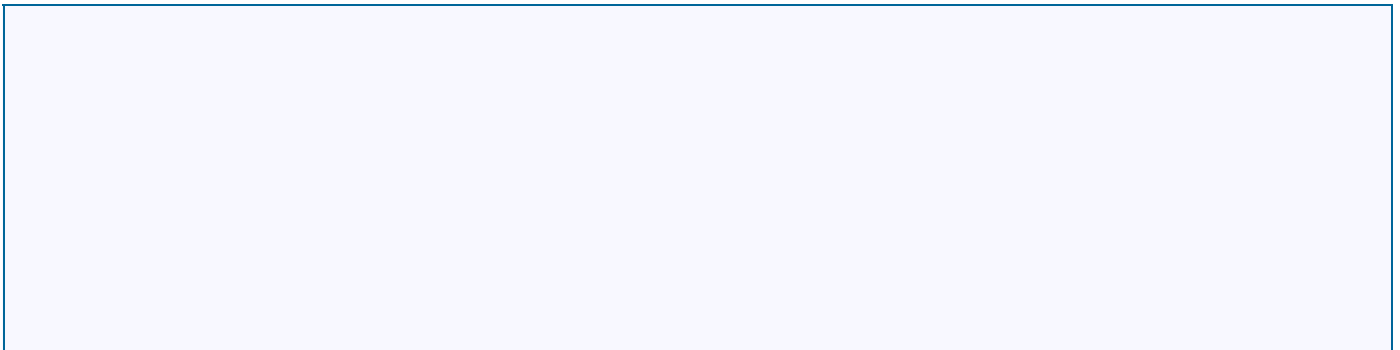
A Network Address List is specified as multi-line text data.

Each text line should contain one of the following:

- one IP address
- an address range - two IP addresses separated with the minus (-) symbol: a range includes both IP addresses and all addresses between them
- an address and a numeric mask, separated with the slash (/) symbol.
  The mask value should be between 1 and 32 for IPv4 addresses and between 1 and 128 for IPv6 addresses. It specifies how many higher bits of the specified address are valid. The remaining lower bits of the address must be zero. The range includes all addresses with the specified higher bits.

The first IP address can be preceded with the exclamation point (!) symbol. In this case the specified IP address or the address range is excluded from the list composed using the preceding lines.

A comment (separated with the semicolon (;) symbol) the can be placed at the end of a line.
Lines starting with the semicolon symbol, and empty lines are comment lines.

---

# LAN Addresses

If you use CommuniGate Pro in a corporate environment, most of your users will connect to the Server from the corporate LAN(s).

Use the WebAdmin Interface to specify your LAN Addresses. Open the Network pages in the Settings realm, then open the LAN IPs page.

| LAN IP Addresses |
|---|
|  |
| Server LAN IP Address:  disabled |
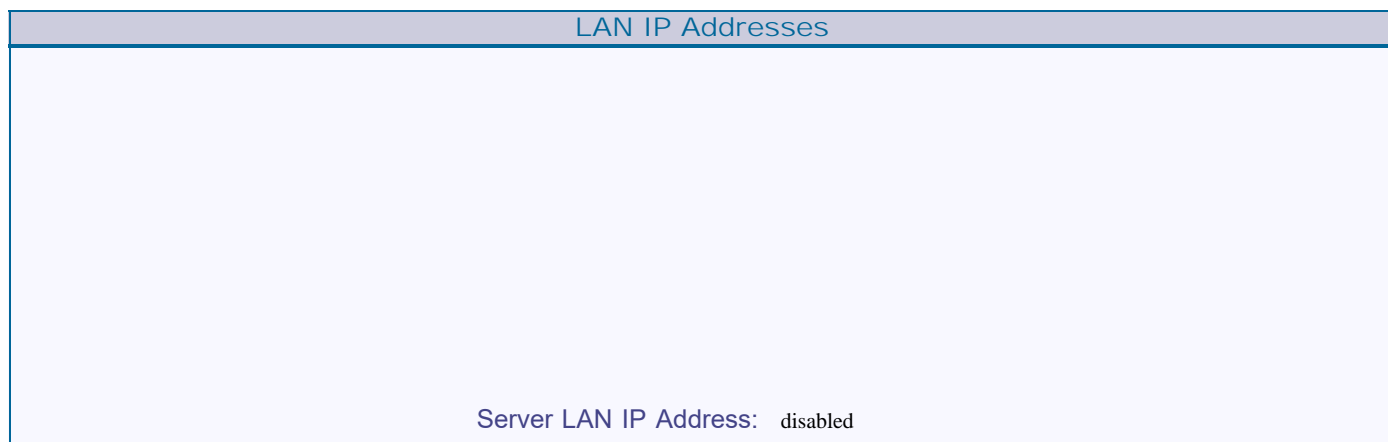
The LAN IP Addresses table initially contains the addresses the CommuniGate Pro software retrieved from the Server OS configuration. Correct this list to include all LAN (local networks) the CommuniGate Pro Server needs to serve. The Network Address Lists section explains the list format.

Usually, you want all E-mail and Real-Time (VoIP/IM) clients connecting from the LAN addresses to be able to relay E-mails and Signals to any Internet destination. In this case you may want to include the LAN addresses into the Client IP Addresses list.

The list of LAN IP Addresses is used to support Real-Time (voice, video, etc.) communications, so the CommuniGate Pro Server knows which addresses are belong to NAT'ed ("local") addresses, i.e. which addresses cannot be contacted directly from the Internet.

Use the Server LAN IP Address setting to select the Server own IP Address the Server OS uses to communicate with computers on the LAN.

---

# Port Allocation

CommuniGate Pro can let the OS select "ephemeral ports" for outgoing TCP connections, or it can allocate these ports itself.
The ports used for Media Proxies and active FTP data connections are always allocated by the CommuniGate Pro Server itself.

Use the WebAdmin Interface to specify the port allocation parameters. Open the Network pages in the Settings realm, then open the LAN IPs page.

| Port Allocation | | | |
|---|---|---|---|
| UDP: | - | | Round-Robin Allocation |
| TCP: | - | Reusage Delay: 30 sec | Use for Media Proxy only |

UDP

> This setting specifies the port number range to be used for UDP proxy operations. If the CommuniGate Pro Server is behind a NAT/Firewall, make sure that all UDP packets received by the NAT/Firewall for these ports are relayed to the CommuniGate Pro Server.

TCP Ports

> This setting specifies the port number range to be used for outgoing TCP connections, including proxy operations. If the CommuniGate Pro server is behind a NAT/Firewall, make sure that all TCP connections received by the NAT/Firewall for these ports are relayed to the CommuniGate Pro Server.

Round-Robin Allocation

> When this option is selected, UDP and TCP ports are allocated evenly using the entire port range.
> When this option is not selected, UDP and TCP ports are allocated using the first (lowest) available port in the port range.

Reusage Delay

> An explicitly allocated TCP port can be re-used after a pause, which should not be smaller than the OS TCP TIME_WAIT time period. Use this setting to specify the re-usage delay.

Use for Media Proxy only

> When this option is selected, TCP ports are explicitly allocated for TCP media proxy and FTP data channels only.
> When this option is not selected, TCP ports are explicitly allocated for all outgoing TCP connections (SMTP, RPOP, SMPP, etc.)

# NATed Addresses

CommuniGate Pro can provide SIP and XIMSS signaling, and media communications for remote clients located behind NAT devices, implementing the *far-end NAT traversal* functionality.

See the NAT section for more details.

# NAT/Firewall Parameters

There are two main types of LAN installations:

- your CommuniGate Pro Server is installed behind a NAT/Firewall device; or
- your CommuniGate Pro Server has at least two network interfaces, one connected to the LAN, and one - to the Internet (WAN).

| Local NAT/Firewall/Load Balancer | |
|---|---|
| WAN IPv4 Address: | WAN IPv6 Address: |

`WAN IPv4 Address`

If your CommuniGate Pro Server has several network interfaces, some connecting it to the LAN, and some - to the WAN (Internet), use this setting to specify the IP address the Server OS uses by default when connecting to remote hosts over the Internet:
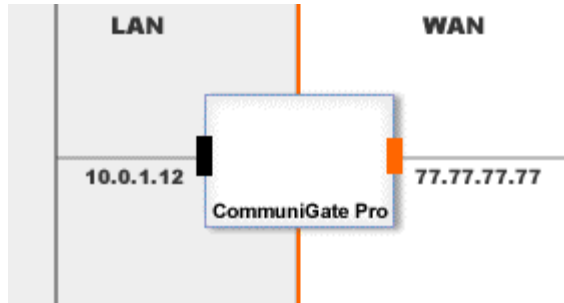


If your CommuniGate Pro Server is installed on a LAN behind a NAT/Firewall, the NAT/Firewall device should be configured to relay all connections on its communication (POP, SMTP, SIP, XMPP, etc.) ports to the CommuniGate Pro Server LAN address. Use this setting to specify the IP address your NAT/Firewall "relays" to CommuniGate Pro.

For example, if your CommuniGate Pro Server has the 10.0.1.12 IP address on your LAN, and the NAT/Firewall relays all incoming connections coming to the 77.77.77.77 IP address to the 10.0.1.12 address, specify the 77.77.77.77 IP address in this setting:
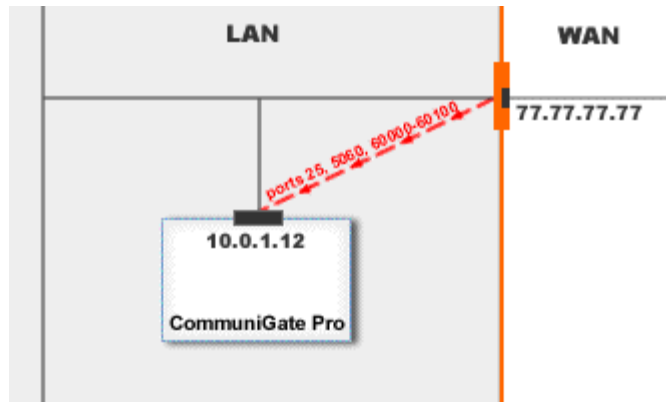


`WAN IPv6 Address`

If your CommuniGate Pro Server is connected to the IPv6 network, specify the Server IP address the Server OS uses by default to connect to remote hosts over the IPv6 Internet.

# Domain Name Resolver (DNR)

Use the WebAdmin Interface to configure the Resolver settings. Open the Network pages in the Settings realm, and follow the DNS Resolver link.

| Domain Name Resolver | | | |
|---|---|---|---|
| Log Level: | Problems | Concurrent Requests: | 10 |
| DNS Servers Addresses: | OS-specified | Source IP Address: | OS default : |
| Balance Server Load: | Disabled | Use Supplementary Responses: | Enabled |
| Initial Time-out: | 7 sec | Retry Limit: | 4 |

Log Level

Use this setting to specify what kind of information the Domain Name Resolver should put in the Server Log. Usually you should use the `Major` or `Problems` levels. In the later case you will see the information about all failed DNS lookups. If you use the RBL services, you may see a lot of failed lookups in the Log. When you experience problems with the Domain Name Resolver, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well. The Resolver records in the System Log are marked with the `DNR` tag.

Concurrent Requests

This setting limits the number of concurrent requests the Resolver can send to Domain Name Servers. On a heavily-loaded Mail or Signal relay processing many thousand requests per second, this parameter should be selected after some testing: older DNS servers may crash if requested to process too many concurrent requests.

DNS Addresses

This setting specifies how the CommuniGate Pro Server selects the DNS servers to use. If the `OS-specified` option is selected, the Server reads the DNS server addresses from the Server host OS. To force the server to re-read those addresses, click the Refresh button on the General page in the Settings section. If the `Custom` option is selected, the CommuniGate Pro Server will use the DNS servers addresses listed in the text field next to this pop-up menu. If no DNS server address is specified, the CommuniGate Pro Server uses the 127.0.0.1 address, trying to connect to a DNS server that can be running on the same computer as the CommuniGate Pro Server.

Balance Server Load

If this option is disabled, then the initial request is always sent to the first DNS server in the list, and if there is no response from that server, the request is resent to the second DNS server, etc. If this option is enabled, then initial requests are sent to different DNS servers: the first initial request is sent to the first DNS server (and if it fails - the request is resent to the second DNS server), the second initial request is sent to the second DNS server (and if it fails - the request is resent to the third DNS server), etc. Enable this option if your Server performs a lot of DNR operations and it can use several equally effective DNS servers.

Use Supplementary Responses

If this option is enabled, then "supplementary" records in MX and SRV requests are processed. These records contain the IP addresses (A and AAAA records) for the domain names listed in an MX or SRV response, so no additional DNR request is needed to retrieve these IP addresses.

Initial Time-out

The Domain Name System uses the connectionless UDP protocol by default, and if there any network trouble, a UDP request or response can be lost (while the TCP protocol automatically resends lost packets). The Domain Name Resolver waits for a response from a DNS server for the period of time specified with this option. If a response is not received, the Resolver resends the request, and waits twice longer, if it times out again, it can resend the request again and wait three times longer. If you have several Domain Name Servers specified, each time the resolver needs to repeat a request, it sends it to the next DNS server in the list.

**Retry Limit**
> This option specifies how many time the Resolver should re-send the same request if it has not received any response from a DNS server.
> **Note:** when a request is an [RBL](#) request, the Resolver sends the same request not more than twice, and both times it uses the same (Initial) response time-out.

**Source IP Address**
> This option selects the *source network address and port* for UDP DNR requests send. If an IP address and/or a port is not specified, the address and/or port is selected using the Server host OS.

The Domain Name Resolver uses TCP connections if a DNS server sent a UDP response with the "Truncated" flag set. This feature allows the Resolver to retrieve very large records from DNS servers.

| Dummy IP Addresses |
|:---:|
|  |

**Dummy IP Addresses**
> This [Address List](#) setting allows you to specify network (IP) addresses that should be considered as "non-existent".
>
> Some DNS authorities may choose to "map" all non-existent names within their domains to some special IP address(es).
>
> When a domain name is resolved into IP addresses, the Resolver checks the first address. If this address is listed in the Dummy IP Addresses list, the Resolver returns the "unknown host/domain name" error code.

The Domain Name Resolver caches responses to SRV-type DNS requests.

| Cache | |
|:---|---:|
| Limit:  30 | Cache Negative:  10 min |

**Limit**
> The maximum Cache size. When the number of items in the cache exceeds this limit, the oldest unused records are being removed from the cache.

**Cache Negative**
> Use this setting to specify for how long negative (failure) DNS responses should be cached.

---

# IPv6 Support

The CommuniGate Pro Server provides full support for both IPv4 and IPv6 network protocols for the following Server Operating Systems:

- Linux
- FreeBSD
- MacOS X
- Windows (Vista and newer)
- Solaris

- NetBSD
- HP/UX
- AIX
- Tru64

If the Server runs on a platform with IPv6 support, and it detects any local IPv6 address, it assumes that the IPv6 networking is enabled.
In this case, the Server creates all network sockets as IPv6 sockets. These sockets communicate with IPv4 systems using the *IPv4-mapped IPv6 address* method.

**Note:** The *IPv4-mapped IPv6 address* method is disabled by default in FreeBSD and NetBSD system kernels. Use the

```
sysctl -w net.inet6.ip6.v6only=0
```

command in your OS startup scripts to enable this method.

**Note:** The *IPv4-mapped IPv6 address* method is permanently disabled in OpenBSD system kernels. As a result, IPv6 networking is not supported on this platform.

You can explicitly instruct the Server to switch IPv6 networking support on or off by using the `--IPv6` [Command Line Option](#):

- `--IPv6 NO` switches the IPv6 support off, even if some local IPv6 addresses are detected.
- `--IPv6 YES` switches the IPv6 support on, even if no local IPv6 local address is detected, but the Server OS supports IPv6 networking.

---

# Denied Addresses

You may want to deny access to your Server for all incoming TCP connections and UDP packets coming from certain IP Addresses.

Use the WebAdmin Interface to specify the Denied Addresses. Open the Network pages in the Settings realm, then open the Blacklisted IPs page.

| Denied IP Addresses |
|---|
|  |
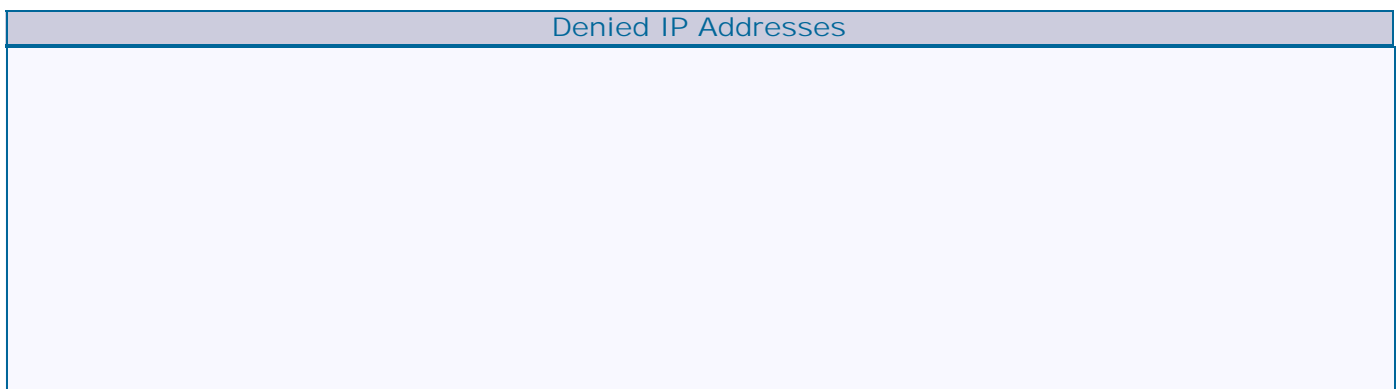
The TCP and UDP [Listeners](#) consult with this IP Address list before they check their own restrictions settings.

In a [Cluster](#) environment, connections and packets from an IP Address are denied if that Address is included into either Server-wide or Cluster-wide Denied IP Addresses list.

---

# Debug Addresses

You may need to obtain a detailed Log of all communications with certain clients or remote servers.

Use the WebAdmin Interface to specify the Debug Addresses. Open the Network pages in the Settings realm, then open the Debug IPs page.

| Debug IP Addresses |
|---|
|  |
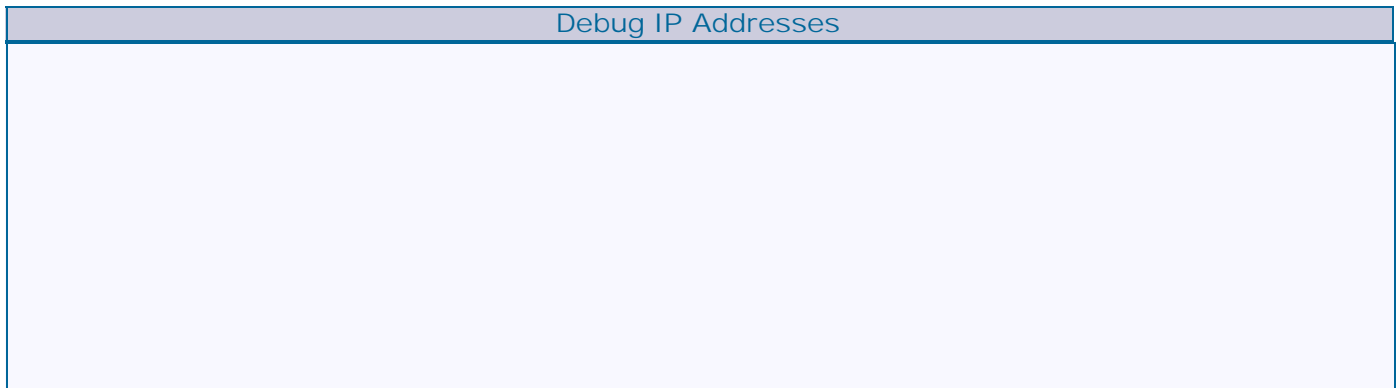
When the Server:

- accepts a TCP connection from an address in this list
- opens a TCP connection to an address in this list
- receives a UDP packet from an address in this list
- sends a UDP packet to an address in this list

the protocol Log Level for that connection or packet is set to All Info.

In a Cluster environment, both the Server-wide or Cluster-wide Debug Addresses lists are checked.

| Network | **Listeners** | Dialup |
|---|---|---|

# Listeners

- **Multi-Socket Listening**
- **Restrictions**
- **Secure Sockets**
- **Limiting Connections from the same Network Address**
- **Reserving Connections for Client Addresses**

The CommuniGate Pro Server accepts SMTP, IMAP, POP, LDAP, and other TCP connections using TCP *Listeners*.
The CommuniGate Pro Server receives SIP, SNMP, RADIUS, TFTP and other UDP requests using UDP *Listeners*.

A Listener opens one or several *listener sockets*, each socket accepting TCP connections or receiving UDP packets directed to the specified port number and, optionally, to the specified local IP address.

The Listener settings allow the Server administrator to specify remote address restrictions, providing access to the listener sockets only to the specified networks.

The CommuniGate Pro TCP Listeners can be configured to accept *clear text* connections or *secure* (TLS/SSL) connections.

The CommuniGate Pro TCP Listeners can limit the number of incoming connections that come from the same IP address. This can help to prevent certain Denial of Service (DoS) attacks.

# Multi-Socket Listening

TCP and UDP services use Listeners to accept incoming connections and packets. Each Listener can use one or several *listener sockets*.

A listener socket accepts incoming TCP connections or receives UDP packets on the port number you specify. You should also specify if the socket should accept connections or receive packets on all IP addresses your server computer has, or only on a selected address.

For example, you may want to create a socket that accepts all connections on one local IP address, while the other socket is used to accept connections on the other local IP address - and only from the specified networks.

Because of the nature of TCP sockets, you cannot have two listener sockets that use the same port number and the same local IP address: if you create a listener socket on the TCP port N that works with ALL local IP addresses, you cannot create a different socket on the same port N. If you create a listener socket on the TCP port N and a specific local address xx.xx.xx.xx, then you can create a different listener socket on the same TCP port N and a different local address yy.yy.yy.yy.

If your CommuniGate Pro Server coexists with some other server software, such as a third-party Web server, you may

want to configure that Web server to use one local IP address, while your CommuniGate Pro server would provide its HTTP services on a different local IP address - but on the same port number. If that port number is 80, and the domain name `www.company.com` resolves into the first IP address, while `mail.company.com` resolves into the second IP address, then typing `http://www.company.com` in a client browser will bring up the third-party Web Server home page, while typing `http://mail.company.com` will bring up the CommuniGate Pro Login page - with both servers running on the same server computer.

| Port | Local IP Address | Init SSL/TLS | Remote IP Address Restrictions |
|------|------------------|--------------|-------------------------------|
|      | all addresses    | off          | Grant                         |
|      | all addresses    | on           | None                          |
|      | all addresses    | off          | None                          |

To create a new listener socket, change the value in the last table element from 0 to the desired port number and click the Update button.

To remove a listener socket, change its port number to 0 and click the Update button.

Even if your server has only one IP address, you may want to create two listener sockets for most of your TCP services: one for regular, *clear text* connections and one (on a different port number) for secure connections (see below).

# Restrictions

You may want a listener socket to accept connections or to receive packets only from certain remote network (IP) addresses.

If the remote IP address of an incoming connection or a received packet is included into the [Denied IP Addresses](#) list, the connection is rejected and the packet is dropped.

If you set the Restriction setting to `Grant`, and list the IP addresses in the text field, the socket will accept connections or packets from the specified addresses only.

If you set the Restriction setting to `Deny`, and list the IP addresses in the text field, the socket will deny access to all clients trying to connect from the specified (blacklisted) addresses.

The IP addresses are specified in a multi-line format. See the [Network](#) section for more details.

There is a difference between the Access Restrictions on the listener socket level, and the restrictions set in the SMTP module. When a remote site connects to your server SMTP port and the site IP address is not accepted by the listener socket, the connection is closed immediately. As a result, the remote site will try all other IP addresses of your server, and then it will try to relay mail via your back-up server.

On the other hand, if the remote site address is included into the Server [Protection](#) Black-List, SMTP sessions are not

closed immediately. Instead, the SMTP session starts and the remote (blacklisted) server is allowed to send the addresses of message recipients. But the SMTP module rejects each address with a "fatal error" code, thus stopping the blacklisted host from trying to relay those messages via your back-up servers.

There is a difference between the Access Restrictions on the listener socket level, and the restrictions set by the Grant Access to the Clients Only option. When a remote site connects to your Server POP, IMAP, WebUser, or other access-type port and the site IP address is not accepted by the listener socket, the connection is closed immediately. As a result, the remote site may try all other IP addresses of your server (and you may have different access restrictions on listener sockets serving those addresses).

On the other hand, if the remote site address is not included into the Server Client IP Addresses list, sessions are not closed immediately. Instead, an access-type session starts, and, if the Grant Access to Clients Only option is enabled, an error message is sent to the remote site before the module closes the connection.

# Secure Sockets

The `Init SSL/TLS` option is available for TCP Listeners only.

Set the `Init SSL/TLS` listener socket option to `On` to tell the Listener component to initiate SSL/TLS negotiations as soon as a connection from a remote site is accepted. Only when a secure connection is established, the Listener allows the communication module to initiate its own protocol (IMAP, HTTP, etc.) - on top of the secure SSL/TLS protocol.

**Note:** Please read the Security section and configure your Domain TLS certificates before you set this option to `On`.

**Note:** When a Listener accepts a connection on a Secure Socket, it tries to detect the CommuniGate Pro Domain the client has connected to. At this time no information has yet been transferred from the client to the server, so the local server IP address the client has connected to is the only data CommuniGate Pro can use to detect the target Domain. If you want a Domain to have its own Security Certificate and to use it for Secure Socket connections, that Domain **must** have an IP address assigned to it.

When the Domain is selected, the Listener retrieves the Domain Certificate and initiates a secure (SSL/TLS) session. If the selected Domain does not have a Certificate, the connection is dropped and an error message is placed into the CommuniGate Pro Log.

**Note:** The current versions of the Internet protocols support the STARTTLS/STLS or equivalent commands. These commands are used to provide secure communications **without** creating a special Secure Socket on an additional port. Instead, a regular port is used, and a regular, non-secure connection is established, and then the client sends the STARTTLS or an equivalent command, and the client and server initiate the SSL/TLS session. If the software you use employs the STARTTLS command (as most SMTP software packages do these days), then you do not need to create any special Secure Socket for secure (SSL/TLS) communications.

Set the `Init SSL/TLS` listener socket option to `Ext` to tell the Listener component that all connections coming to this socket are SSL/TLS secured, but that there is an external device implementing all SSL/TLS encryption/decryption operations.
Connections coming to these ports are clear-text connections, but higher-level CommuniGate Pro components and protocols process these connections as if they come encrypted: clear-text Login operations are considered secure, STARTTLS operations are prohibited, etc.

# Limiting Connections from the same Network Address

The connection-limiting options are available for TCP Listeners only.

To prevent various Denial of Service (DoS) attacks you may want to limit the number of connections a Listener can accept (on all sockets) from the same IP address:

| Reserve Connections for Clients: | 10 | Limit Connections from same Address: | 10 | Non-Clients: | 10 |
|---|---|---|---|---|---|

The Non-Client setting applies only to the TCP connections from Network IP Addresses not included into the Client IP Addresses list. This setting value should be equal or smaller than the main Limit Connections setting value.

When you set these settings, you should remember that:

- IMAP clients usually open several connections to the server. If you set this setting for the IMAP listener to less than 5, you can cause problems for your users.
- Web browsers can open several simultaneous connections to retrieve embedded graphic files and other HTML page elements.
- Many Web clients can connect to your server via the same proxy, and they all appear as connecting to your server from the same IP address.

**Note:** to avoid problems with inter-server communication, this setting has no effect on connections that come from other Servers in the same CommuniGate Pro Cluster.

# Reserving Connections for Client Addresses

To prevent various Denial of Service (DoS) attacks you may want to set aside a certain number of available TCP connections for those who connect to your Server from Client IP Addresses.

If the difference between the maximum number of allowed connections and the number of active connections is equal to or smaller than the specified Reserve value, connections from non-Client IP Addresses are rejected.

CommuniGate **Pro**

| Introduction |
| Installation |
| SysAdmin |
| **Network** |
| Objects |
| Storage |
| E-mail |
| Real-Time |
| Access |
| Services |
| Directory |
| Clusters |
| Applications |
| Miscellaneous |
| Licensing |

| WebMail |
| Pronto! |
| PBX |

| Network | Listeners | **Dialup** |

# Dialup

- **Mail Receiving**
- **TCP Activity Schedule**
- **Serving LAN Clients**

The CommuniGate Pro Server can work on a site that only has a dial-up Internet connection. It allows LAN users to access their mail and to submit messages without generating any Internet TCP traffic, and it allows the System Administrator to specify the Schedule for Internet (dialup) TCP/IP activity.

# Mail Receiving

Dial-up systems are not connected to the Internet all the time. As a result, most of the time other systems cannot send mail directly to your dial-up server.

You can use three methods to receive incoming E-mail messages:

- store all mail in a Unified Domain-Wide Account on your ISP server, and retrieve it from there periodically, using the RPOP module.
- specify your ISP mail server as your mail backup server, and use the SMTP *Remote Queue Starting (ETRN)* feature to retrieve mail using the SMTP module.
- specify your ISP mail server as your mail server (main MX), and use the SMTP *ATRN* feature to retrieve mail using the SMTP module. The ISP mail server should be able to handle the ATRN requests. If the ISP server is a CommuniGate Pro Server, too, it should be configured to hold mail for your Domain and to accept ATRN commands from your Server.

Your Server should have a *static IP* address to be able to receive mail via SMTP (using ETRN). If your ISP assigns you an IP address dynamically, and each time your Server can get a different IP address, retrieving mail using the SMTP ATRN module or the RPOP module are the only choices.

# TCP Activity Schedule

The Server Administrator can specify when and how often the Server is allowed to generate outgoing TCP/IP traffic. This helps to limit the time your Internet dial-up link is up.

The TCP Activity Schedule is checked within the [SMTP](#) and [RPOP](#) modules and can be used to limit their activities.

Use the WebAdmin Interface to configure the TCP Activity Schedule. Open the Network pages in the Settings realm, and follow the Schedule link.

| IP Activity Schedule | | | |
|---|---|---|---|
| Log Level: Failures | | | |
| Day(s) of Week | When | | Pause |
| Every Day | 08:00 - | 18:00 | 5 minutes |
| Every Day | 18:00 - | 08:00 | hour |
| Never | midnight - | midnight | 0 seconds |

`Log Level`
> Use this setting to specify what kind of information the TCP Activity Schedule component should put in the Server Log. Usually you should use the `Major` (session starts) levels. But when you experience problems with the TCP Activity Schedule component, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case the schedule calls and schedule processing details will be recorded in the System Log as well.
>
> The TCP Activity Schedule component System Log records are marked with the `TCP` tag.

`Day of Week`
> This setting specifies the week days when this TCP Scheduler element should be used.

`When`
> This setting specifies the time period when outgoing TCP Activity is allowed.
>
> If the first time setting is larger than the second one, it specifies an "over-the-midnight" time period: `19:30 - 07:30` means from 19:30 till midnight and from midnight till 7:30 in the morning.

`Pause`
> This setting specifies the minimal time interval between successive outgoing "TCP sessions".

You can remove elements from the Schedule by setting the Day of Week option to `Never`, and you add elements to the Schedule by changing the Day of Week option value in the last dummy (`Never`) element.

---

# Serving LAN Clients

Your LAN client mailers can generate outgoing Internet activity when they submit messages via SMTP, and this can force your dial-up link to go up every time they send a message.

To avoid this problem:

- make sure that your own CommuniGate Pro Server, not the ISP mail server is specified in the client mailer

settings as the "outgoing mail server";
- make sure that IP addresses of all your LAN Clients are included into the Client Hosts list;
- make sure that SMTP module [Verify Return-Path for](#) option is set to `non-clients` or `nobody`.

**Objects** Domains Accounts Groups Forwarders Named Tasks Chronos

# Objects

- **Domains**
- **Domain Objects**
- **Accounts**
- **Groups**
- **Forwarders**
- **Account Aliases**
- **Mailing Lists**
- **Named Tasks**
- **Database**

Each CommuniGate Pro Server has a hierarchy of *objects* it serves:

- on the topmost level there is a set of Domains
- each Domain contains Accounts, Groups, Mailing Lists, Aliases, Forwarders and Named Tasks
- Each Account contains one or several Mailboxes
- Each Mailbox contains some number of messages

Besides these basic objects, CommuniGate Pro supports supplementary objects: Accounts can contain File Storage and Preference Data, Domains can contain Certificates, WebUser Skin files, Real-Time Applications, etc.

# Domains

Domains are the CommuniGate Pro objects that contain other objects: Accounts, Mailing Lists, Groups, Aliases, Forwarders and Named Tasks. Each Domain has a domain name (client.com, www.company1.com, etc.):
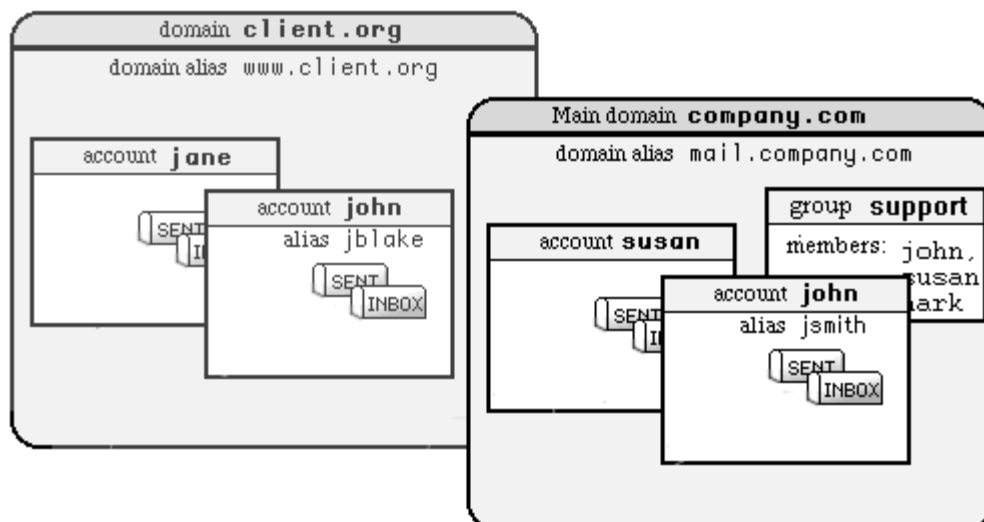
While each CommuniGate Pro Domain has its domain name, it is not necessary to create a separate CommuniGate Pro Domain for each domain name you want to serve. CommuniGate Pro Domains can have Domain Aliases: they allow you to assign several names to the same CommuniGate Pro Domain. For example, the CommuniGate Pro Domain `company.com` may have a Domain Alias `mail.company.com`. In this case all references to the domain name mail.company.com will be processed as references to the company.com CommuniGate Pro Domain.

There is a special CommuniGate Pro Domain, called the *Main Domain*. Other CommuniGate Pro Domains are called *secondary domains*. The Main Domain is created as soon as the Server is installed, and its name is specified in the General Settings. If your Server should serve only one Domain, the Main Domain is all you need and there is no need to create secondary domains. The Main Domain name is used as the Server Name.

Each CommuniGate Pro Domain has its own settings and a set of Domain Objects.

See the Domains section for more information about CommuniGate Pro Domains.

# Domain Objects

Each Domain has its own, independent set of Objects: Accounts, Groups, Forwarders, Aliases, Mailing Lists, Named Tasks. Each Object should have a name that is unique within the Domain. Different Objects in different Domains can have the same names.

Object names are case-insensitive. Object names can contain Latin letters, digits, the underscore (_), the minus (-), and the point (.) symbols. The point symbol cannot be used as the first or the last symbol of an Object name.

Object names should not contain more than 128 symbols.

Use the WebAdmin Interface to view Domain Objects. Open the Users realm, and follow the link for the selected Domain.
To open the Domain object list, you should have the All Domains access right or a Domain Administrator right for this Domain.

If you are a Domain Administrator, then the list of Objects in your Domain appears on the main Domain Administration page.

| 1000 | Filter: | | | | |
|------|---------|---|---|---|---|
| Accounts (2 of 345) | Account Details | Forwarders (1 of 10) | Aliases (0 of 5) | Mailing Lists (1 of 1) | |
| **Object** | **Type** | | | | |
| lsmith | MultiMailbox | 2 | | | |
| smith-L | Group | | *15* | | |
| support | Mailing List | | 18 | | |
| susan | Forwarder | | susan@otherdomain.dom | | |
| xsmith | Text Mailbox | 34K | 20:34:56 [10.0.8.195] | | 1 |
| x.smith | Alias | | *xsmith* | | |

To select Objects by name, enter a string into the Filter field, and click the Display button: only the Objects with names containing the specified string will be displayed.

The pop-up menu allows you to limit the number of Objects to be displayed.

The checkbox options specify the type of Objects you want to display: Account, Groups, Forwarders, Aliases. The information will include the selected and total number of those Objects in the Domain.

Each line in the list contains an Object name and its type.

- If the Object is an Account, the Account type is displayed.
  If the Account Details option is selected, the line displays the information about:
    - the Mail storage used with this Account
    - the last time when the Account was opened and the network address from which it was accessed
    - the current number of real-time Devices registered with this Account
  This Account Details retrieval operation is resource-consuming: do not use it if you want to display a lot of Accounts on one page.
- If the Object is an Alias, the real Account name is displayed.
- If the Object is a Forwarder, the forwarding address is displayed.
- If the Object is a Group, the number of Group members is displayed.
- If the Object is a Mailing List, the number of List subscribes is displayed, but only if at least one List distribution has been performed.

# Accounts

An Account is the basic *service unit*: every user served with a CommuniGate Pro Server should have an Account in one of the Server [Domains](#).

Each Account is protected with a password, so only the Account owner (and, optionally, System and Domain Administrators) can have unrestricted access to Account data.

When the CommuniGate Pro Server is installed, the `postmaster` Account is automatically created in the Main Domain.
The [Master](#) (unlimited) access right is granted to that Account.

Account E-mail and Signal address is

*accountname@domainname*
where *accountname* is a name of a CommuniGate Pro Account, and *domainname* is the name of the CommuniGate Pro Domain in which this Account is created.

Messages directed to an Account address are delivered to the Account using the [Local Delivery](#) module.

Signals directed to an Account address are delivered using the [Signal](#) component.

An administrator can create [Account Aliases](#) to assign multiple names to one Account.

Each CommuniGate Pro Account has its own settings and [Storage](#) for the Account Mailboxes, files, and other data.

See the [Accounts](#) section for more information about CommuniGate Pro Accounts.

# Groups

CommuniGate Pro Domains can contain Groups. Groups are essentially lists of Account names and/or other groups and sending a message to a group results in sending it to all group members.

See the [Groups](#) section for more information about CommuniGate Pro groups.

# Forwarders

CommuniGate Pro Domains can contain Forwarders. Each Forwarder has a name and contains an E-mail address for redirection. If mail is sent to *name@domain.com* where *name* is a Forwarder object in the domain.com CommuniGate Pro Domain, then mail is re-routed to the E-mail address specified in that Forwarder object.

Group and Forwarder Objects are different:

- a Forwarder can contain only one address, while a group can contain several addresses;
- a Forwarder works on the [Router](#) level, substituting its own address with the specified address, while a group object actually processes a message sent to the group and generates a new message copy to be sent to the group members.

See the [Forwarders](#) section for more information about CommuniGate Pro Forwarders.

---

# Account Aliases

An Account Alias is an alternative name assigned to a CommuniGate Pro Account. Each Account can have zero, one, or several Account Aliases.

For example, the Account `j.smith` in the `domain2.com` Domain can have aliases `smith` and `jsmith`. Mail sent to the `smith@domain2.com` address will be stored in the `j.smith` Account, and attempts to login as `jsmith@domain2.com` will open the same `j.smith` Account.

You can use [Forwarders](#) to assign alternative name for Accounts, too. If you create the Forwarder `js` in the `domain2.com` Domain, and make it point to the `j.smith` address, it will work as yet another alias for the `j.smith` Account.

If you rename the Account `j.smith` into `james.smith`, all Account Aliases will "move" with it - `smith` and `jsmith` will remain the Aliases for the `james.smith` Account. If you remove the Account, the Account Aliases will be removed, too.

Renaming and removing of Accounts has no effect on the Forwarders: if you rename or remove the `j.smith` Account, the Forwarder `js` will continue to point to the `j.smith` address.

As a result, it is not recommended to use Forwarders where you can use Aliases. Forwarders should be used to create "objects" that redirect mail to other Domains or to other mail servers.

---

# Mailing Lists

CommuniGate Pro Domains can contain Mailing Lists. Each Mailing Lists has a name and it always belongs to some Account in the same Domain - the Mailing List owner.

A Mailing List contains a list of subscribers, and it maintains several Mailboxes in the list owner Account. Those Mailboxes are used to store and archive postings, generate digests, store subscription requests and error reports.

Groups and Mailing Lists are different:

- Groups are designed for a small number of members, Mailing Lists are designed to handle several hundred thousand subscribers per list;
- Groups are designed mostly for local members (Accounts) and if the subscriber Account is renamed or removed,

it is also renamed in or removed from all the Groups in its Domain;
- Mailing Lists provide a lot of features beyond basic mail distribution: automatic subscribing, bounce processing, archiving and digesting, browsing, posting policies, moderating, etc.

See the LIST section for more information about CommuniGate Pro Mailing Lists.

# Named Tasks

CommuniGate Pro Domains can contain Named Tasks. Each Named Task has a name and it always belongs to some Account in the same Domain - the Named Task owner.

A Named Task is a Real-Time Application task that is automatically started with the Server when a Signal or an E-mail is addressed to the Named Task name. All those Signals and E-mails are delivered to a single instance of the Real-Time Application task, even in the Cluster environment.

Named Tasks are used to implement collaboration mechanisms such as Instant Message "chat rooms", various communication gateways, etc.

See the Named Tasks section for more information about CommuniGate Pro Named Tasks.

# Database

## Domain Files

All CommuniGate Pro Domain data is stored inside a file directory created in the `Domains` subdirectory inside the Server *base directory*. This directory name is the same as the Domain name.
The Main Domain data is stored inside the `Accounts` file directory created inside the *base directory*.

Inside a Domain file directory, a `Settings` file directory is created. This directory contains files with the domain-wide data:

Access.settings
> This file has the dictionary format, it contains the names of the users that have administrative access rights to the Server or to this Domain, and the list of the granted rights. By storing all administrative access rights in one location the CommuniGate Pro Server makes it easier to maintain server security. Only the `Access.settings` file stored in the Main Domain `Settings` directory can contain the server-level administration access rights. All other `Access.settings` files can contain only the Domain-level Administration access rights.

Domain.settings
> This file contains the Domain Settings.

Template.settings
> This file contains the Account Template for this Domain and provides the Account Settings for new Accounts in this Domain.

Aliases.data
> This file contains the list of all account-level aliases specified for the Domain Accounts.

LISTS
> This directory contains files with the information about the mailing lists created in the Domain.

WebSkins
> This directory contains custom WebUser Interface Skins for this Domain.

```
PBXApp
```
This directory contains custom PBX Application Environment for this Domain.

# Account Service Files

Every CommuniGate Pro Account contains at least one (INBOX) Mailbox, and at least two service files. Service files have the following file name extensions:

`.settings`  this dictionary file contains Account Settings, including the Account Rules.

`.info`  this dictionary file contains volatile Account information, such as Mailbox sizes, SIP Registrations, Event subscriptions, etc.
Since the `.info` file is being modified rather often, the CommuniGate Pro Server is built to survive .info file corruptions. For example, if the Mailbox last UID information is corrupted, the Server rescans the Mailbox and restores the correct Mailbox info.

`.dst`  this optional dictionary file contains the Account "DataSet", that includes Account DataSet Address Books (also known as string lists), and application preferences set via ACAP.

`.web`  this optional file directory is the Account File Storage.

`.balances`  this optional file directory contains files with Billing history information.

# Account Files Location

The Account files are located in the Domain file directory or in its subdirectory (see the Domains section for the details). The `GetAccountLocation` CLI command can be used to learn the physical location of the Account files.

For a multi-mailbox Account, a directory with the Account name and `.macnt` extension is created, and all Account files are stored in that directory. The Account service files are stored as `account.extension`. The INBOX Mailbox is stored as the `INBOX.mailboxType` file.
Example: for the multi-mailbox Account `John`, the `john.macnt` directory is created, and the files `INBOX.mbox`, `account.settings`, `account.info` are placed in that directory.

For a single-mailbox Account, the INBOX Mailbox is created as a file in the Domain file directory or its subdirectory, and it has the `accountName.mailboxType` file name. The Account service files are stored in the same directory as `accountName.extension`.
Example: for the single-mailbox Account `John`, the `john.mbox`, `john.settings`, and `john.info` are placed into the Domain file directory.

Objects **Domains** Accounts Groups Forwarders Named Tasks Chronos

# Domains

- **Displaying the Domain List**
- **Creating a New Domain**
- **Specifying Domain Settings**
- **Default Domain Settings**
- **Multihoming and Dedicated IP Addresses**
- **Client IP Addresses**
- **Enabling Services**
- **Domain Limits**
- **Domain Aliases**
- **Directory Integration**
- **Server OS Integration**
    - **Legacy (Unix) Mailer Compatibility**
- **Domain Security Settings**
- **Provisioning**
- **Processing Unknown Names**
- **Sending Mail To All Accounts in the Domain**
- **Sending Mail To All Accounts in All Domains**
- **WebUser Interface Settings**
- **SMTP Options**
- **Domain Rules**
- **Administrator Domain**
- **Renaming Domains**
- **Removing Domains**
- **Suspending Domains**
- **Domain File Directories**
    - **Domain Subdirectories**
    - **Storage Mount Points**
    - **Account Subdirectories in Large Domains**

CommuniGate Pro Server can serve Accounts, Goups, Mailing Lists, Forwarders and other objects in its Main Domain, and, optionally, in multiple Secondary Domains, each with its own set of Accounts (and other objects such as Mailing Lists, Groups, and Forwarders).

Every Domain can have zero, one, or several *Domain Aliases* (alternative names). All Domain names and Domain Aliases should be unique, and they should be registered with the Domain Name System (DNS).

In many cases, a Domain should not have a separate set of user Accounts, but should rather be a domain name alias for an already existing CommuniGate Pro Domain. You may also want to serve some domain names using account mapping and/or Unified Domain-Wide Accounts.

In all these cases, you do not have to create a new CommuniGate Pro Domain to serve a domain name.

When a client application connects to your CommuniGate Pro Server, and specifies an account name, the Server has to decide in which Domain to look for that Account. See the Access section for the details.

# Displaying the Domain List

Use the WebAdmin Interface to view the list of Domains served with your Server. Open the Domains page in the Users realm.

To open the Users realm, you should be connected as the `postmaster` or any other Server Administrator with the [All Domains](#) access right.

| 10 | Filter: | | | **Telephone Numbers** | | | | |
|---|---|---|---|---|---|---|---|---|
| Domains: 4 of 4 | | Domain Aliases: 3 of 3 | | Accounts: 13172 of 13172 | | | | |
| **Domain** | **IP Address** | **Accounts** | **Open** | **Hits** | **Last Hit** | **Refs** | | |
| client1.com | 192.0.0.2 | 645 | 24 | 2891 | 21:29:17 | 14 | Settings | |
| client2.com | | 78 | 5 | 3456 | 21:30:32 | 7 | Settings | |
| **mycompany.com** | 192.0.0.1 | 1380 | 89 | 15890 | 21:30:29 | 35 | Settings | |
| mail.client1.com | | *client1.com* | | | | | Settings | |
| mail.client1.com | | *client2.com* | | | | | Settings | |
| webmail.client1.com | | *client1.com* | | | | | Settings | |

To select Domains by name, type a string into the `Filter` field, and click the Display button: only the Domains with names containing the specified string will be displayed.

Each entry in the Domain list contains the Domain name, the assigned network address (if any), and the number of Accounts in the Domain. If the Domain is a shared Domain served by a [Dynamic Cluster](#), the Domain name has the `[+]` prefix. If the Domain is a Directory-based Domain, its name is displayed with the `[D]` prefix.

A list entry also displays the number of currently opened Domain Accounts, the total number of times the Domain Accounts have been opened (since the Server last restart), and the last time any Domain Account was opened.

Select the `Show Aliases` option to include Domain Aliases into the list. Each Domain Alias element contains the link to its "real" Domain object list and settings pages.

Click a Domain name to view the [Objects](#) in that Domain.

Click the word Settings in the last column to view and update the Domain Settings.

Click the Telephone Numbers (Telnum) link to view all [Telnums](#).

---

# Creating a New Domain

Type a new Domain name into the field on the right side of the Create Domain button.

| |
|---|
| *Created* |

Click the Create Domain button. When a new Domain is created, its name appears in the Domain List.

If this Server is a member of a [Dynamic Cluster](#), the additional Create Dynamic Cluster Domain button appears. Click that button to create a Domain that will be served with all Cluster members. The Domain created using the Create Domain button are created as "local" Domains and are served with this Server only.

International (non-Latin) symbols are not allowed in the Domain names, but they are allowed in the [Domain Alias](#) names.

# Specifying Domain Settings

The Main Domain and all Secondary Domains have Domain-level settings.

To open the Domain Settings page in your browser, either click the Domain Settings link in the Domains List, or click the Domain Settings link on the Domain Object list page.

| Comment |
| --- |
| |

The Comment field allows you to enter arbitrary information about the domain.

| Account Log Level: | Problems | Mailbox Log Level: | Problems |
| --- | --- | --- | --- |

The Account Log option allows you to specify how the account-level operations (account open/close, password verifications, Mailbox creating/removing, size updates, etc.) are recorded. Log records created for account-related events have the `ACCOUNT` tag.

The Mailbox Log option allows you to specify how the mailbox-level operations (message storing/removing, message status updating, etc.) are recorded. Log records created for Mailbox-related events have the `MAILBOX` tag.

Most of Domain Settings can be set to the `default` value. In this case the actual setting value is taken from the global, Server-wide Default Domain Settings.

When the Domain Settings are modified, click the Update button. The page should appear again, displaying the *Updated* marker.

You can click the Objects link to switch to the Domain Object List.

---

# Specifying Default Domain Settings

A Domain setting can have the `default` value. In this case the actual setting value is taken from the server-wide Default Domain Settings. You can modify these Default values by clicking the Domain Defaults link on the Domains (Domain List) page.

The Default Domain Settings page resembles a regular Domain Settings page.

A Dynamic Cluster installation maintains separate server-wide Default Domain settings for all non-Shared (Local) Domains, and cluster-wide Default Domain settings for all Shared Domains. In the Cluster environment, the Default Domain Settings page displays links that allow you to switch between the Server-wide and Cluster-wide Default Settings.
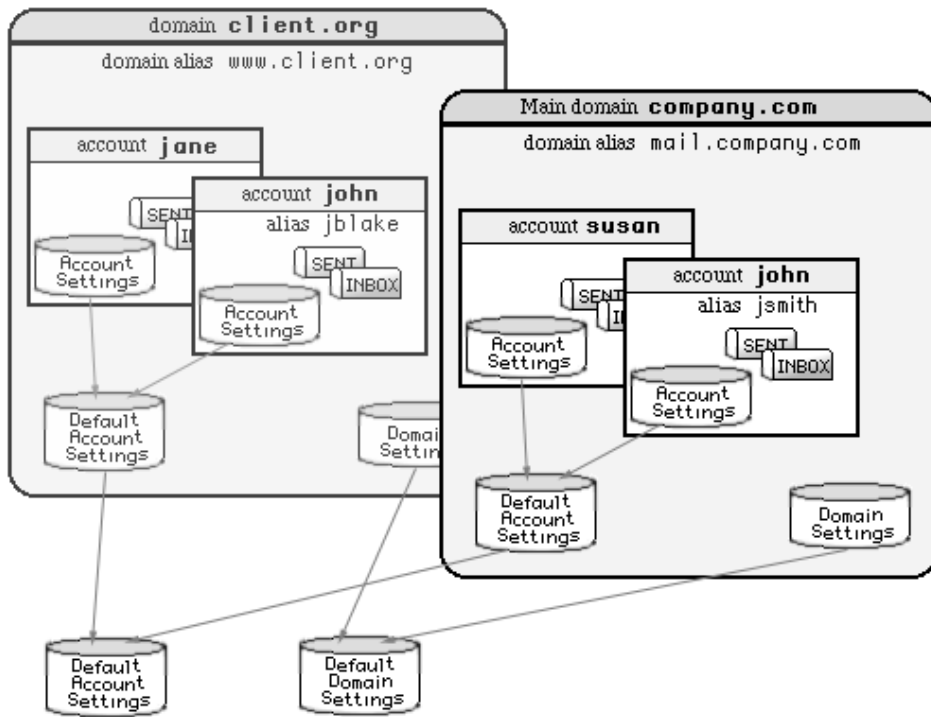
# Multihoming and Dedicated IP Addresses

You should read this chapter only if you plan to support multihoming, if your system is behind a firewall, or if you have a non-standard Domain Name System setup.

When the Server starts, it detects its own network address(es). Your Server system is "multihomed" if it has more than one network (IP) address.

If the Server system has several IP addresses, some of them can be assigned (dedicated) to secondary Domains. Accounts in such Domains can be accessed using any POP, IMAP, or other client application without explicitly specifying the full Account name.

The Assigned IP Addresses option allows you to assign network addresses to the main and secondary Domains.

| Assigned IP Addresses | |
|---|---|
| All Available | [206.40.74.198] |
| Use for Outgoing Connections: | default(No) |

`All Available`
> This option can be selected for one Domain only, and it is the default setting for the Main Domain.
> All Server's network addresses not assigned to other Domains are assigned to this Domain.

`Manually Defined`
> This option is selected by default for all secondary Domains.
> If you want to assign (dedicate) an IP address to this Domain, type the address into the text field on the right of the pop-up menu. You can specify several IP addresses, separating them with the comma ( , ) symbol.
> Only the Server computer's own addresses are accepted, and all specified addresses should not be already assigned to any other CommuniGate Pro Domain.
> If you select this option and leave the text field blank, the Domain will not have any IP addressed assigned to it. In this case, to access the Domain Accounts, users should specify the full Account name (*account@domain*) in their client application settings. See the Access section for the details.

413

```
by DNS A-Record
```
When this option is selected, the Server sends a request to the Domain Name System and tries to resolve the Domain name. If an A-Record for this CommuniGate Pro Domain is found in the Domain Name System, the addresses from that record are assigned to the Domain. The system checks that all addresses retrieved from the A-record belong to the Server computer and that these addresses have not been already assigned to any other Domain.

This setting is useful if you have several secondary Domains with dedicated IP addresses and you want to redistribute the Server addresses from time to time. Instead of reconfiguring both DNS and Server settings, you may reconfigure the DNS records only, and the Server will take the updated data from the DNS.

```
by DNS MX-Record
```
When this option is selected, the Server retrieves the highest-priority MX record (relay name) for this CommuniGate Pro Domain, and then processes addresses in the A-record for that relay name.

```
Use for Outgoing Connections
```
If this option is enabled, then the non-LAN IP addresses assigned to this Domain (if any) can be used as source network address when establishing outgoing connections (such as SMTP, RPOP, XMPP, etc.)

For each Domain in the Domain List, the assigned network (IP) addresses are displayed. This can be used to check the DNS and Server setup for systems with multihoming.

Because of setup errors or due to a non-standard network and DNS setup, the Server's own IP address(es) may be left unassigned to any of the Server domains. Open the General Settings page to see the list of the Server own IP addresses. The unassigned addresses are marked in red.

When a client application connects to the Server via an unassigned address and the full account name is not specified, the Server does not allow the user to log in.

# Client IP Addresses

Each Domain can have its own list of Client IP Addresses, which extends the Server-wide or Cluster-wide Client IP Addresses list for this Domain Account users:

| Client IP Addresses |
|---|
|  |

# Enabling Services

Each Domain has a set of settings that specify which CommuniGate Pro services can be used with the Domain Accounts. See the Accounts section for the details.

| Enabled Domain Services | | | | | | | |
|---|---|---|---|---|---|---|---|
| Default | Mail | Relay | Signal | Mobile | TLS | POP | IMAP | MAPI |
| | AirSync | SIP | XMPP | WebMail | XIMSS | FTP | ACAP | PWD |
| | LDAP | RADIUS | S/MIME | WebCAL | WebSite | PBX | HTTP | |

Services can also be disabled for individual Domain Accounts.

A service is available for an Account only if that service is enabled for the Account itself AND for the Account Domain. Disabling a service in the Domain Settings disables that service for *all* Domain Accounts.
**Note:** This is different from disabling a service in the Domain Default Account Settings: disabling a service in the Default Account Settings disables that service only for those Domain Accounts that have the Enabled Services option set to `default`.

# Domain Limits

The System Administrator can specify some limits on the resources available to the Domain users.

A Domain Administrator can see, but cannot modify these limits.

| Resources | Limits | Usage |
|---|:---:|---|
| Accounts: | 10000 | 390 |
| Mailing Lists: | 5 | 5 |

# Domain Aliases

Each CommuniGate Pro Domain can have aliases (alternative names). If the `client.dom` Domain has the `mail.client.dom` and `www.client.dom` Domain Aliases, E-mail and Signals directed to user@mail.client.dom and to user@www.client.dom will be routed to the user@client.dom Account. Also, to access the user@client.dom Account via POP, IMAP, XMPP, and other client applications the Account names user@mail.client.dom and user@www.client.dom can be specified in the client settings.

This is especially useful for [WebUser](#) clients. Users specify the domain name in their browser URLs, and users of the `client.dom` Domain tend to use `www.client.dom` in the browser URLs. You may want to register the www.client.dom domain name with the DNS, assigning it the same IP address as the address assigned to the client.dom Domain, and then you should create the www.client.dom Domain Alias for the client.dom Domain.

| Domain Aliases |
|---|
| |

You can modify existing Domain Aliases, add an Alias by typing a new name in the empty field, and remove an Alias by deleting it from its field. Use the Update button to update the Domain Aliases list.

The Domain Aliases are stored in the `DomainAliases` [database](#) located in the Settings directory inside the CommuniGate Pro *base directory*.

# Directory Integration

The System Administrator can specify if the Domain Accounts should be included into the Central Directory.

| Directory Integration | |
|---|---|
| Object Records: | Keep In Sync |

This panel is not displayed for Directory-Based Domains, since those domains are always completely integrated with the [Directory](#).

See the [Directory Integration](#) section for the details.

# Server OS Integration

CommuniGate Pro Accounts may be "mapped" to the accounts (registered users) of the Server OS. See the [Accounts](#) section for more details.

## Legacy (Unix) Mailer Compatibility

The CommuniGate Pro allows you to create Accounts with [Legacy INBOX](#) Mailboxes. These Mailboxes are stored not inside the CommuniGate Pro *base directory*, but in the system file directory known to the legacy mailer applications.

If you have to support local mailer compatibility for all or some Accounts in this Domain, you should specify the Legacy INBOX settings:

| Server OS Integration | |
|---|---|
| Legacy INBOX location: | ~ |
| Legacy INBOX locking used: | File Level |

`Legacy INBOX location`
> This setting specifies where the Legacy INBOX files should be located. For each Account that has a Legacy INBOX, the Server substitutes the asterisk (`*`) symbol with the CommuniGate Pro Account name.
> Consult with your OS manuals to see where your legacy mailers expect to find user Mailboxes. On most Unix systems, the `/var/mail/` directory is the correct location, but some systems may use `/var/spool/mail/` or some other directory.

`Legacy INBOX locking used`
> This setting specifies the file locking method to use for updates synchronization.

See the [Mailboxes](#) section for the details.

# Domain Security Settings

A Domain can have its own set of enabled Authentication methods. See the [Security](#) section for more details.

A Domain can have PKI settings (Private Keys and Certificates) enabling secure communications ([TLS](#), [Certificate Authentication](#), [S/MIME](#)) with that Domain.
Use the Security link on the Domain Settings page to open the Domain Security settings.
See the [PKI](#) section for more details.

A Domain can be configured to add `DKIM-Signature` headers to outgoing messages.
Use the Security link on the Domain Settings page to open the DKIM settings.
See the [PKI](#) section for more details.

A Domain can have Kerberos keys enabling "secure single sign-on" for that Domain.
Use the Security link on the Domain Settings page to open the Domain Security settings.
See the [Security](#) section for more details.

# Provisioning

The Domain objects (Accounts, Groups, Forwarders, etc.) should have unique names within that Domain.
Each name should contain from 1 to 250 latin letters, decimal digits, the dot ( . ), underscore ( _ ), minus ( - ) symbols.
A name should not start or end with the dot ( . ) symbol.
A name should not be the same as one of the Special Address names.

You can control how the Server creates, renames, and removes Domain Accounts.

| Account Provisioning | |
|---|---|
| Auto-Signup: | default(Disabled) |
| Consult External on Provision: | Yes |
| Auto-Create Chatrooms: | default(Disabled) |

Auto-Signup

If this option is enabled, users can create Domain Accounts themselves, via the WebUser Interface, or by using XMPP, or XIMSS clients.

If this option is enabled, the Sign-up link appears on the WebUser Interface Domain Login page, and the XIMSS and XMPP modules report their *self-registration* capabilities.

The Server checks that no Account with the specified name exists and creates a new Account.
The Server uses the Account Template settings for the newly created Account, overriding its Password and Real Name settings with the data specified by the new user.

Consult External on Provision

If this option is enabled, the CommuniGate Pro Server sends a pair of commands to the External Authentication Helper application every time a Domain Account is created, renamed, or removed, when the Account License Class is changes. One command is sent before the Server performs the operation, and the other command is sent after the Server successfully completes the operation.

Auto-Create Chatrooms

If this option is enabled, users can create chatroom Named Tasks by trying to connect to a non-existent chatroom in this Domain. See the Named Tasks section for the details.

---

# Processing Unknown Names

Addresses used in E-mail messages, in client "login names", and in Signals can contain unknown names. If the Server cannot find an Object (an Account, a Mailing List, an Alias, a Group, or a Forwarder) with the specified name, the Domain Unknown Names settings are used.

| Unknown Names | | |
|---|---|---|
| Consult External for Unknown: | Yes | |
| Mail to Unknown | Rerouted to | postmaster@client1.com |
| Calls to Unknown: | default(Rejected) | pbx |
| Access to Unknown: | default(Rejected) | error |

Consult External for Unknown

When an unknown name is supplied and this option is enabled, the CommuniGate Pro Server sends a command to the External Authentication Helper application. That application can check an external database (or any other data source)

and optionally create a new object (an Account, an Alias, etc.) with the specified name. If the program returns a positive response, the Server makes one more attempt to find a Domain object.

Mail to Unknown

This setting specifies what the Server should do when unknown account/object names are encountered in E-mail message addresses.

Rejected

The address is rejected; if the message is being received via SMTP, the address is not accepted, and if it was the only message recipient address, the message is not received at all.

Discarded

The address is routed to NULL. The message is considered "delivered" immediately (it is discarded).

Rerouted to:

the address is changed to the E-mail address specified in the text field, and the Router restarts trying to route this new address.

**Note:** you specify an E-mail address, not an account name there. So, if you specify Rerouted To: Postmaster for the client1.com Domain, messages sent to unknown names will be routed to the Postmaster account in the Main Domain, not to the postmaster Account in that client1.com Secondary Domain. Specify postmaster@client1.com to direct those messages to the postmaster Account in the client1.com Domain.

**Note:** you can use the asterisk (*) symbol in the E-mail address field. This symbol will be replaced with the original (unknown) name.

Sample:

The Domain client1.com Mail to Unknown Name option is set to

Rerouted to: Bad-*@support.company.com

A message comes addressed to jjones@client1.com, and the Account jjones does not exist in the client1.com Domain.

The message is rerouted to bad-jjones@support.company.com

Accepted and Bounced

The Router accepts E-mail addresses with unknown names, routing them to the Local Delivery module. When the message is enqueued into the Local Delivery module queue, the module fails to find the addressed account/object, the message is rejected, and an error report is sent back to the sender.

Calls to Unknown

This setting specifies what the Server should do when unknown account/object names are encountered in Signal addresses. This setting is set in the same way as the Mail to Unknown setting.

Access to Unknown

This setting specifies what the Server should do when unknown account/object names are encountered in Access operations. This setting is set in the same way as the Mail to Unknown setting.

---

# Sending Mail To All Accounts in the Domain

The administrator can enable the special virtual list (address) "all" that can be used to send messages to all Accounts created in this Domain.

| <all@company.com> | |
|---|---|
| Mail to All is distributed for: | Authenticated Users |
| Mail to All is sent to Forwarders: | default(No) |

Messages sent to the `<all@`*`domainname`*`>` address are stored directly in the Account INBOX Mailboxes, bypassing any Account [Rules](#).

Messages sent to the `<all@`*`domainname`*`>` address are not stored in the Accounts that have the Accept Mail to All setting disabled.

Mail access to the `<all@`*`domainname`*`>` address can be restricted.

`anybody`
> Any message sent to the `<all@`*`domainname`*`>` is distributed to all Accounts in this Domain.

`Clients`
> A message sent to the `<all@`*`domainname`*`>` address is distributed only if it has been received via SMTP from an Internet address included into the [Client IP Addresses](#) list, or if the message was received using one of the *authenticated* methods (WebUser Interface, XIMSS, via RPOP, via POP using the XTND XMIT method, etc.)

`Authenticated Users`
> A message sent to the `<all@`*`domainname`*`>` address is distributed only if it has been received from a Server user (Account) using one of the *trusted* methods.

`Authenticated Domain Users`
> A message sent to the `<all@`*`domainname`*`>` address is distributed only if it has been received (using one of the *trusted* methods) from an Account in this Domain or from any other Server Account that has the [Domain Administration](#) right for this domain.

`Authenticated Administrator`
> A message sent to the `<all@`*`domainname`*`>` address is distributed only if it has been received (using one of the *trusted* methods) from a Server Account that has the [Domain Administration](#) rights for this domain.

`nobody`
> The `<all@`*`domainname`*`>` address is disabled. In this case it is possible to create the real Account, Forwarder, Group, or Mailing List with the `All` name.

Messages to `<all@`*`domainname`*`>` can be sent to all Forwarder addresses, too:

`Send to Forwarders:`
> When this option is enabled, a new message is composed. Its envelope contains the addresses from all Forwarder objects in this Domain. The message body is a copy of the message sent to the `<all@`*`domainname`*`>`.

# Sending Mail To All Accounts in All Domains

If the administrator has enabled mail distribution to all Accounts in the Main Domain, a message can be sent to all Accounts in all Domains.

To send a message to all Accounts in all server Domains, it should be sent to the `alldomains@`*`main_domain_name`* address.

For each Domain, the message source is checked and the message is distributed to the Domain Accounts only if it passes that Domain "Mail to All" distribution checks.

# WebUser Interface Settings

Each Domain has several [WebUser Interface](#) settings:

| WebUser Interface | |
|---|---|
| Mail Trailer: | |

| | |
|---|---|
| Account Web Site Prefix: | ~ |
| Account Web Site Banner: | |
| Recommended XIMSS Mode: | default(TCP) |

Mail Trailer Text
> The text in this field is appended (optionally) to all messages the Domain users compose via the WebUser and MAPI Interfaces.

Site Prefix
> This option allows you to change the URL prefix needed to access Account File Storage via HTTP. It can be set to an empty string, so URLs for a File Storage will look like `http://domainName:port/accountName/`.
> See the [HTTP](#) module description for more details.

Web Banner Text
> When the HTTP protocol is used to retrieve an HTML file from an Account [File Storage](#) in this Domain, this text is inserted into the beginning of the file HTML "body".

Recommended XIMSS Mode
> This option controls what the `features` [XIMSS Pre-Login](#) response. The following values are supported:

> TCP
>> use a TCP connection for XIMSS sessions, connecting to the same port the current request was sent to.

> XIMSS TCP
>> use a TCP connection for XIMSS sessions, connecting to the first port specified for the [XIMSS Module](#) Listener.

> HTTP
>> use [HTTP Binding](#) for XIMSS sessions.

---

# SMTP Options

The SMTP panel controls how E-mail messages are sent from and received for Accounts in this Domain.

| SMTP | |
|---|---|
| Force AUTH for: | non-clients |
| Force STARTTLS for: | nobody |
| Check Recipient Account: | Disabled |
| HELO Prefix: | mail |

Force AUTH for
> If this option is enabled and the SMTP module receives a Return-Path address that belongs to this Domain, the address

and the message itself are rejected unless the client application user has been authenticated. See the [SMTP Module](#) section for the details.

Force STARTTLS for
> If this option is enabled and the SMTP module receives a Return-Path address that belongs to this Domain, the address and the message itself are rejected unless the SMTP connection secured using SSL/TLS encryption.
> See the [SMTP Module](#) section for the details.

Check Recipient Account
> This option specifies if the [SMTP Module](#) should perform additional checks when processing the RCPT TO command targeting a local Account.
> If this option is enabled, the module accepts the command only if:

> - the Account `Mail` Service is enabled, and
> - the Account Message Storage quota is not exceeded, and
> - the Account Incoming Flow control limits are not exceeded.

HELO Prefix
> When an outgoing SMTP connection is made on behalf of this Domain, the domain name in the HELO/EHLO command contains this Domain name. **Note:** this requires that a non-LAN IP is assigned to the Domain.
> This setting specifies if the Domain name must be prefixed with some subdomain: if the `example.dom` Domain has this setting value set to `mail`, the HELO/EHLO command will contain the `mail.example.dom` parameter.

# Domain Rules

Domains can have Automated Rules that are applied to all E-Mail Messages and all Signals being delivered to Accounts in those Domains. See the [Rules](#) section for more details.

# Administrator Domain

Domains can be controlled by the [Server Administrators](#) and by the [Domain Administrators](#) - Accounts in the same Domain that are granted some Domain Administrator Access Rights. You may choose to grant administration rights for this Domain to Domain Administrators created in a different Domain. In this case the name of that other Domain should be entered into the Administrator Domain Name field:

| Administrator Domain |
| --- |
|  |

If this field is not empty, the Domain Administrator Accounts created in this Domain and the Domain Administrator Accounts created in the specified Domain can be used to administer this Domain.

See the [System Administrator](#) section for more details.

# Renaming Domains

If you want to rename a Secondary Domain, open its Domain Settings page, fill the New Domain Name field, and click the Rename Domain button.

If there is no other Domain with the same name as the specified new domain name, the Domain is renamed and its Domain Settings page should reappear on the screen under the new name.

You cannot rename a Domain when any of its Accounts is in use.

# Removing Domains

If you want to remove a Secondary Domain, open its Domain Settings page, and click the Remove Domain button. The confirmation page appears. If the Empty Domains Only option is selected, a Secondary Domain is removed only if there are no Accounts in it. Otherwise, all Domain Accounts are permanently removed, too.

If you confirm the action, the selected Domain, its settings, all its Accounts and other Objects will be permanently removed.

You cannot remove a Domain when any of its Accounts is in use.

# Suspending Domains

You may want to suspend a secondary Domain to close all its currently open Accounts, sessions, and connections. Attempts to open an Account in a suspended Domain are rejected with a temporary error (and incoming mail is delayed).

Suspend a Domain if you want to perform OS-level maintenance tasks on the Domain storage and you need to ensure that the CommuniGate Pro Server or Cluster is not accessing that storage.

To suspend a Domain, open its Domain Settings page, and click the Suspend Domain button. The Button changes to become the Resume button.

To resume a Domain, open its Domain Settings page, and click the Resume button.

Suspended Domains have the `Suspended` marker on the WebAdmin Domains list page, and their Domain Settings pages have the same marker on the page top.

# Domain File Directories

The Main Domain data is stored in the `Accounts` file directory inside the CommuniGate Pro *base directory*.

The secondary Domains data is stored in the `Domains` file directory inside the *base directory*. For each secondary Domain, a directory with the Domain name is created in the `Domains` directory. All shared Domains in a Dynamic Cluster and stored as subdirectories of the `SharedDomains` directory.

Each Domain directory contains data for all Domain Accounts.

When a Domain contains many Accounts, [Account Subdirectories](#) inside the Domain directory can be used.

## Domain Subdirectories

When a CommuniGate Pro system serves many Domains (more than 3,000), you may want to place Domain files directories into several subdirectories:

- many operating and file systems have limits on the number of files in one directory;
- subdirectories can speed up the Domain and Account files access operations;
- subdirectories can be moved to additional storage devices.

Domain subdirectories are directories inside the Domains or SharedDomains directory. A subdirectory name has the `.sub` file path extension (suffix).

Subdirectories can be nested.

**Note:** When the CommuniGate Pro Server starts, it scans the Domains directory and all its `.sub` subdirectories, and it collects

the names and file paths of all Domains it finds there.

This feature allows the administrator to change the foldering method (see below) without stopping the Server and without relocating already created Domains. It also allows the system administrator to move Domains between subdirectories at any time when the CommuniGate Pro Server is stopped.

When a new Domain is being created (or when an existing Domain is being renamed), the Server composes a name for the subdirectory in which the Domain files should be created. The Domain Storage panel contains the settings that control how a subdirectory name is composed. Open the Domains page of the WebAdmin Interface, and follow the Domain Defaults link to open the page that contains the Domain Storage panel:

| Domain Storage | |
| --- | --- |
| Foldering Method: | default(Flat) |
| Rename In Place: | default(No) |

Foldering Method
> This option allows you to specify the subdirectory name construction method. The following methods are supported:

> flat
>> This is the default method. All new Domains are placed into the Domains directory itself (or SharedDomains directory if a shared Domain is being created or renamed in a Dynamic Cluster).

> 2 Letters 1 Level
>> The first two letters of the Domain name are used to form the name of the subdirectory, the Domain `client1.com` will be placed into the `Domains/cl.sub/` subdirectory. If the Domain name has just one letter, that letter is used as the subdirectory name.

> 2 Letters 2 Levels
>> The first two letters of the Domain name are used to form the name of a nested subdirectory, the Domain `client1.com` will be placed into the `Domains/c.sub/l.sub/` subdirectory. If the Domain name has just one letter, that letter is used as the subdirectory name.

> Hashed 1 Level
>> A numeric hash function is applied to the Domain name, the result is used to form a subdirectory name: the Domain `client1.com` will be placed into the `Domains/nu.sub/` subdirectory.

> Hashed 2 Levels
>> A numeric hash function is applied to the Domain name, the result is used to form a nested subdirectory name: the Domain `client1.com` will be placed into the `Domains/pj.sub/v.sub/` subdirectory.

Rename in Place
> If this option is not enabled, and you rename a Domain, the CommuniGate Pro Server uses the currently set Foldering method to compose a new file path for the renamed Domain and moves the Domain data there. If you have replaced the `xx.sub` directories with symbolic links to directories on different disk volumes, such a rename operation may require moving data from one volume to a different one, and it will fail. If you enable this option, the CommuniGate Pro Server will move (rename) the renamed Domain data within the same directory, so the "cross-volume link" problem will be avoided.

## Account Subdirectories in Large Domains

When a CommuniGate Pro Domain contains many Accounts (more than 10,000), you may want to place account files in several subdirectories:

- many operating and file systems have limits on the number of files in one directory;
- subdirectories can speed up the account files access operations;
- subdirectories can be moved to additional storage devices.

Account subdirectories are directories inside the Domain directory. A subdirectory name has the `.sub` file path extension (suffix).

Subdirectories can be nested.

**Note:** When the CommuniGate Pro Server starts, it scans all Domain file directories and all their subdirectories, and it collects the names of all Domain Accounts. This feature allows the system administrator to move Accounts between subdirectories at any time when the server is stopped. It also allows you to change the foldering method (see below) without stopping the Server and without relocating already created Accounts.

For each Account, the CommuniGate Pro Server remembers the name of the subdirectory that contains the Account files.

When a new Account is being created (or when an existing Account is being renamed), the Server composes a name for the subdirectory in which the Account files should be created.

| Account Storage | |
|---|---|
| Foldering Method: | default(Flat) |
| Rename In Place: | default(No) |
| Generate Index: | default(No) |
| Fast Storage Type: | default(0) |

Foldering Method
    This option allows you to specify the subdirectory name construction method. The following methods are supported:

    flat
        This is the default method. All new Accounts are placed into the domain directory itself.

    2 Letters 1 Level
        The first two letters of the Account name are used to form the name of the subdirectory, the Account `jsmith` will be placed into the `domain/js.sub/` subdirectory. If the Account name has just one letter, that letter is used as the subdirectory name.

    2 Letters 2 Levels
        The first two letters of the Account name are used to form the name of a nested subdirectory, the Account `jsmith` will be placed into the `domain/j.sub/s.sub/` subdirectory. If the Account name has just one letter, that letter is used as the subdirectory name.

    Hashed 1 Level
        A numeric hash function is applied to the Account name, the result is used to form a subdirectory name: the Account `jsmith` will be placed into the `domain/pf.sub/` subdirectory.

    Hashed 2 Levels
        A numeric hash function is applied to the Account name, the result is used to form a nested subdirectory name: the Account `jsmith` will be placed into the `domain/lu.sub/y.sub/` subdirectory.

Rename in Place
    If this option is not enabled, and you rename an Account, the CommuniGate Pro Server uses the currently set Foldering method to compose a new file path for the renamed Account and moves the account data there. If you have replaced the `xx.sub` directories with symbolic links to directories on different disk volumes, such a rename operation may require moving data from one volume to a different one, and it will fail. If you enable this option, the CommuniGate Pro Server will move (rename) the renamed Account data within the same directory, so the "cross-volume link" problem will be avoided.

Generate Index
    If this option is enabled, the CommuniGate Pro Server creates the `Index.data` file in the Domain file directory. This file contains the names of all Domain Accounts, the Account types, and the location of the Account files. When the Server starts and finds the Index.data file in the Domain directory, it reads that file instead of scanning the Domain file

directory tree. On some file systems scanning a directory tree with 100,000 files can take up to 10 minutes.

**Note:** if you have stopped the Server and manually moved/removed some Domain Account directories, delete the Index.data file from the Domain directory before you start the Server again.

**Note:** if you want to keep only symbolic links in the Domain file directory, you can create the `Index` subdirectory inside the Domain directory (or an `Index` symbolic link to some other directory). If this subdirectory exists, the Server stores the `Index.data` file inside that subdirectory rather than in the Domain file directory itself.

```
Fast Storage Type
```
Use this option if you want to store Account "fast" ("settings" and "info") files separately from other Account data. For example, you may want to use smaller, but faster disk storage for these frequently used files.
This option has an effect on multi-mailbox Accounts only.
When this option is set to the zero value, the Account "fast" files are stored within the Account file directory:
`Domains/`*`subdirectory`*`/`*`domainName`*`/`*`subdirectory`*`/`*`accountName`*`.extension/account.`*`fastFileExtension`*
When this option is set to a non-zero value N, the Account "fast" files are stored outside the Account file directory, by inserting the path member `fast` after the Nth path component:
`Domains/`*`subdirectory`*`/`*`domainName`*`/`*`subdirectory`*`/fast/`*`subdirectory`*`/`*`accountName`*`.`*`fastFileExtension`*
For example, if the jsmith@client1.dom Account path is
`Domains/cl.sub/client1.dom/js.sub/jsmith.macnt`
and this option value is 2, the "settings" file path is
`Domains/cl.sub/fast/client1.dom/js.sub/jsmith.settings`
The `Domains/cl.sub/fast` directory can be a link to a separate fast storage volume.

# Storage Mount Points

When a CommuniGate Pro system serves many Domains, especially large Domains, you may want to distribute Domain files and directories between several physical storage volumes.

To create a "Storage mount point", use the CREATEDOMAINSTORAGE CLI command. It creates the `storage name`.mnt directory inside the Domains directory.
Replace that directory with a symbolic link to the selected physical storage volume, or "mount" an additional physical storage volume over this directory.

When at least one "storage mount point" exists, the Create Domain button is accompanied by a pull-down menu listing all available storage mount points. Select a storage mount point to store new Domain files in.

When a Domain is renamed, its files stay within the storage used to create that Domain.

When a CommuniGate Pro Domain has many Accounts, you may want to distribute Account files and directories between several physical storage volumes.

To create a "Storage mount point", use the CREATEACCOUNTSTORAGE CLI command. It creates the `storage name`.mnt directory inside the Domain directory.
Replace that directory with a symbolic link to the selected physical storage volume, or "mount" an additional physical storage volume over this directory.

When at least one "storage mount point" exists, the Create Account button is accompanied by a pull-down menu listing all available storage mount points. Select a storage mount point to store the new Account files in.

When an Account is renamed, its files stay within the storage used to create that Account.

---

| Objects | Domains | **Accounts** | Groups | Forwarders | Named Tasks | Chronos |

# Accounts

- **Creating a New Account**
- **Specifying Account Settings**
  - **Authentication Methods**
  - **Two-factor Authentication**
  - **Enabled Services**
  - **Access Settings**
  - **Mail Storage Settings**
  - **Incoming Mail Transfer Settings**
  - **Outgoing Mail Transfer Settings**
  - **Signaling Settings**
  - **File Storage Settings**
  - **WebUser Interface Settings**
- **Account Aliases**
- **Account Telephone Numbers**
- **Access Rights**
- **Renaming Accounts**
- **Removing Accounts**
- **Default Account Settings**
- **Class of Service**
- **Account Template**
- **Importing User Account Information**

An Account is the basic *service unit*: every user served with a CommuniGate Pro Server should have an Account on that Server.

Each Account is protected with a password, so only the Account owner (and, optionally, System and Domain Administrators) can have access to Account data.

The `postmaster` Account is automatically created in the Main Domain. The Master (unlimited) access right is granted to that Account.

The `pbx` Account is automatically created in the Main Domain. See the PBX section for more details.

# Creating a New Account

To create a new Account, type a new Account name into the field on the right side of the Create Account button and click that button. The selected Account name should meet the Domain Object name restrictions.

|  | **Template** |
|---|---|
| Multi-Mailbox | Legacy INBOX (in `/var/mail/`) |

Use the pop-up menu to specify the Account type:

`MultiMailbox`
A folder-type Account that can contain several Mailboxes of various types, as well as File Storage. The INBOX Mailbox is automatically created within the new Account. All incoming E-mails are stored in the INBOX Mailbox by default. The user can create additional Mailboxes using any IMAP client or AirSync client software, or using the CommuniGate Pro Web E-mail Interface.

`Text INBOX, MailDir INBOX, ...`
An Account containing only a single INBOX Mailbox, and no file storage. You can select any supported Mailbox format.

If the user plans to use just POP3 client software, only one Mailbox is needed, and you may want to create a Single-Mailbox type Account for that user.

By default, the Account name becomes the person's E-mail name, so Account names should contain only letters, digits, dash and point (dot) symbol - some mail systems cannot send mail to E-mail addresses containing other symbols.

`Legacy INBOX (in /var/mail/)`
>  Select this option if you want the new Account INBOX to be created as an [Legacy Mailbox](#), so new Account can be used with legacy *local mailers*. This option is enabled only if the Legacy Mailbox location is specified in the Domain Settings.

Click the Create Account button. When a new Account is created, its name appears in the Domain Objects list. The Server automatically displays the [Settings page](#) for the new Account.

The Settings of a newly created Account are automatically set to the [Account Template](#) values.

You can create several Accounts at once, by preparing an Account List file and using the [Import](#) option.

If you have the Domain Administrator (not the Server Administrator) access rights, you need to be granted the `CanCreateAccounts` access right, and you may also need the `CanCreateSpecialAccounts` access right.

---

# Specifying Account Settings

To specify Account Settings, click the Account name link in the Accounts list. The Account Settings page appears.

| | |
|---|---|
| Created: 12-May-2007 | **Access Rights** |
| Real Name: | |
| Title: | |
| City: | |
| Services: | |
| Organization: | |
| Space Test: | |
| CommuniGate Password: | |
| FirstName: | |
| FamilyName: | |
| department: | |

`Real Name`
>  This field is used to specify the real-life user name. The Server uses this information to compose the default 'From' address in Web Mailer.

additional *System* fields
>  If the Server [Directory Integration](#) settings contain some System Custom Account Setting fields, these fields appear in this panel where they can be set and modified.

`CommuniGate Password`
>  The Account password. When authenticating a user, the Server can check either this password or OS password, or both (see below).

additional *Public Info* fields
>  If the Server [Directory Integration](#) settings contain some Public Info Custom Account Setting fields, these fields appear in this panel where they can be set and modified.

The modified values of the Real Name and additional fields are updated in the Directory if the Domain has the [Directory Integration](#) setting set to Keep In Sync.

After the Account Settings are modified, click the Update button.

# Authentication Methods

Use the Authentication panel to specify the Account authentication methods.

| Authentication | | Secure Only: | default(No) |
|---|---|---|---|
| CommuniGate Password: | default(Enabled) | OS UserName: | * |
| Password Modification: | allow | | |
| Minimal Password Length: | default(6) | OS Password: | default(Disabled) |
| Password Complexity: | default(any) | Authentication URI: | ldap://10.0.1.2:389/uid=*,dc=company,dc=com |
| Password Encryption: | default(A-crpt) | | |
| Ask to change Password every: | 10 day(s) | External Password: | default(Disabled) |
| | | Log Protocols: | default(Disabled) |
| Password Recovery: | default(Enabled) | Log Login/Logout: | default(Disabled) |
| Kerberos Login: | default(Enabled) | Alt RADIUS Password: | |
| Certificate Login: | default(Disabled) | Alt SIP Password: | |
| Failed Logins Limit: | 30  in  20 sec | Last Failed Login: | 05-Aug [64.173.55.170] |

To log into the CommuniGate Pro Account, a user (a client application) supplies a password, directly, or via some secure authentication protocol. For successful authentication:

- the supplied password should match the CommuniGate Password stored in the Account settings (see above), or
- the supplied password should match the password of the "mapped" Server OS account (see below), or
- the supplied password should be verified using Authentication URI (see below), or
- the supplied password should be verified using an External Authenticator program.

Besides the password-based authentication methods, CommuniGate Pro Server can authenticate its Account users using other methods, such as Kerberos and TLS Certificates.
See the [Security](#) section for the details.

```
CommuniGate Password
```
CommuniGate Pro Account internal password can be stored as an Account setting (see above).
The following settings control how this internal CommuniGate password is used.

```
CommuniGate Password
```
This setting tells the Server if it should compare the user-provided password and the internal CommuniGate Password.

```
Password Modification
```
This setting controls if the user can modify the CommuniGate Password via the [PWD module](#), the [WebUser Interface](#), [XMPP](#) module, or the [XIMSS](#) Interface.

```
Password Complexity
```
This setting specifies how complex the user-supplied CommuniGate Password should be.

```
any
```
no restriction

```
mixed case letters
```
the password must contain both upper-case and lower-case letters

```
letters and digits
```
the password must contain both upper-case and lower-case letters, and at least one decimal digit.

```
Password Encryption
```
This setting specifies how the Server should store the CommuniGate Password. If the `clear` option is selected, the password is stored as a clear-text string. All other options specify various encryption methods. In most cases, you will not specify this setting on a per-account basis, but rather using the Domain Account Defaults or global Account Defaults.

```
U-crpt
```

The          password encryption is used for compatibility with the Unix "crypt" encryption method and it should be used for migrating users from other servers only. The U-crtp-encrypted passwords can not be used for Secure (SASL) Authentication methods.
See the [Security section](#) for the details.

`Ask to change Password every`
This setting specifies how often the Server will ask the user to change the password.
User will be asked to change the password on login to the Account via [WebUser Interface](#) or Pronto! client.

`Alt RADIUS Password`, `Alt SIP Password`
These alternative passwords (if set) are used for [RADIUS](#) and [SIP](#) authentication operations.
**Note:** when you set new values for these options, they are stored using the current `Password Encryption` method.

`Password Recovery`
This setting controls if the user can use the [WebUser Interface](#) or the [XIMSS](#) Interface to ask the Server to send the CommuniGate Password to the "recovery E-mail" address.


`Server OS Integration`
CommuniGate Pro Accounts can be "mapped" onto the accounts (registered users) of the Server OS. When a CommuniGate Pro user is being authenticated using a Server OS password, or when a separate process (program) should be launched on the user behalf, the CommuniGate Pro Server constructs an OS *username* (OS account name) to be used for that CommuniGate Pro user (Account).

`OS UserName`
This setting specifies how to compose the Server OS *username*. The asterisk (`*`) symbol is substituted with the CommuniGate Pro Account name. If this setting contains just one symbol - the asterisk symbol, then the CommuniGate Pro Account is "mapped" onto the OS account with the same name: when the CommuniGate Pro Server checks the OS password for the Account `jmsith`, it checks if the specified password can be used to log into the OS account `jsmith`.

If the setting contains `*.dj`, the OS username for the CommuniGate Pro Account `jsmith` is `jsmith.dj` - and the `jsmith.dj` name is used for all OS-level operations initiated on behalf of the CommuniGate Pro Account `jsmith`.


`OS Password`
If this option is enabled, the Account user can log in using the password set in the Server OS registration information for this user.


`Other Login Methods`
CommuniGate Pro Accounts users can use other methods to log into their Accounts.

`Authentication URI`
An External Resource can be used for authentication.
The URI (Uniform Resource Identifier) is of the form `scheme://address[:port]/parameters`

If the value of the `scheme` equals `ldap` or `ldaps` then the Server will make LDAP bind request to `address` using `parameters` as DN (Distinguished Name); and if the External Resource answers positively then the user can log into the Account. Works only with "clear text" authentication methods.
**Note:** for Microsoft Active Directory LDAP server as `parameters` instead of DN you can use `DOMAIN\account` where `DOMAIN` is the short Windows domain name and `account` is the value of `sAMAccountName` attribute of the account record in the Active Directory.
Other values of `scheme` are ignored. In `parameters` the asterisk (*) symbol is substituted with the CommuniGate Pro Account name, the `^0` is substituted with the Domain name.

`Kerberos Login`
If this option is enabled, the user can log into the Account using the Kerberos Authentication method.
See the [Security section](#) for the details.

`Certificate Login`
If this option is enabled, the user can log into the Account using the Client Certificate Authentication method.
See the [PKI section](#) for the details.

`External Password`
If this option is enabled, the user can log into the Account using a password verified with the External Authenticator program.
See the [Security section](#) for the details.

`Secure Only`

This option requires use of [secure authentication methods](#) (APOP or non-clear-text SASL methods) with this Account. If a user client application connects to the Server and supplies a password for this Account using an insecure ("clear text") authentication method, the Server will reject the connection even if the supplied password is correct. Clear-Text password are still accepted if they are passed through a secure (SSL/TLS) communication channel.

If the option is set to `Require TLS` then the login is possible only through connections secured with SSL/TLS.

**Note:** Since OS passwords can be checked only using the clear-text authentication method, enabling the Secure Only option forces the users employing OS passwords to use secure (SSL/TLS) communication channels.

`Log Protocols`

If this option is enabled, access protocol operations (POP, IMAP, ACAP, LDAP, etc.) for this Account are recorded in the System Log using the All Info Log Level.

`Log Login/Logout`

If this option is enabled, login/logout operations for this Account are recorded in the special [Supplementary Log](#).

`Failed Login Limit`

This setting specifies a time period and the number of incorrect login attempts that a user or users can make before the Account is disabled for login operations. Account logins are re-enabled after the same period of time.

The Server remembers the IP Address of the client that performed the last successful login. Logins from this IP Address are not blocked even when the number of incorrect login attempts exceeds the limit.

`Last Failed Login`

This element is displayed when there was a failed attempt to log into this Account. The element displays the attempt date and/or time, and the IP address the attempt was made from.

Click the Clear button to remove the Last Failed Login info and to reset the Account Failed Logins counter.

If the CommuniGate Password, OS Password, Kerberos, Certificate, and External Authentication options are disabled, the user will not be able to access the Account.

Any Authentication setting can be set to the `default` value, in this case the setting value is taken from the Domain Default Account Settings or the Server-wide or Cluster-wide Default Account Settings.

## Two-factor Authentication

CommuniGate Pro supports customizable set of methods of verification that password authentication is performed by a user who possess something that can be contacted separately from the primary authentication channel: for example a mobile phone ready to receive an SMS, a land-line phone ready to receive a call, an e-mail address on another mail system, and so on. Having verified the account's primary password, CommuniGate Pro uses the `x2auth.sppr` PBX script to deliver to the user an one-time password using either of these additional methods. The user then has to enter that one-time password to confirm the possession of the items used for this additional authentication.

This additional authentication is supported for logins through the [WebUser Interface](#) or the [XIMSS](#) protocol. After a successful login the additional verification can be not requested for the logins from the same IP address for the specified period of time.

## Enabled Services

There is a set of settings that specify which CommuniGate Pro services can be used with the Account:

| Enabled Services | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Mail | Relay | Signal | Mobile | TLS | POP | IMAP | MAPI |
| Default | AirSync | SIP | XMPP | WebMail | XIMSS | FTP | ACAP | PWD |
| | LDAP | RADIUS | S/MIME | WebCAL | WebSite | PBX | HTTP | |

`Mail`

If this Service is disabled, incoming E-mail Messages are not delivered to this Account. Incoming messages are suspended in the [Local Delivery](#) Module queue, and they are rejected if this option is not re-enabled within the specified period of time.

See the [Local Delivery](#) module settings for the details.

If this option is disabled, Account cannot compose and submit E-mail Messages.

`Relay`

If this Service is disabled, Account user is not able to use the [Mobile Users Support](#) features.

You may want to disable this Service when you provide free WebMail Accounts, and you do not want spammers to use these Accounts to enable SMTP relaying.

`Signal`
If this Service is disabled, incoming Signals sent to this Account are rejected.

`Mobile`
If this Service is disabled, the Account user cannot connect (login) from Internet Addresses not included into the Client Addresses list. You may want to disable this Service if you want your users to connect to their Accounts only from the Internet Addresses on your own network.

`TLS`
If this Service is disabled, secure (SSL/TLS) access to this Account is disabled.

`POP, IMAP, MAPI, AirSync, PWD, XMPP, FTP, ACAP`
If a protocol Service is disabled, the Account cannot be opened (the Account user cannot be authenticated) using that protocol.

`SIP`
If this Service is disabled, SIP operations requiring authentication (REGISTER, outgoing calls, etc.) are not available for this Account.

`WebMail`
If this Service is disabled, the Account cannot be opened using the WebUser Interface.

`XIMSS`
If this Service is disabled, the Account cannot be opened using the XIMSS Interface.

`SMIME`
If this Service is disabled, Secure Mail (S/MIME) features implemented in the WebUser and XIMSS Interfaces are not available for this Account.

`LDAP`
If this Service is disabled, the Account user cannot be authenticated with the LDAP module.

`RADIUS`
If this Service is disabled, the Account user cannot be authenticated with the RADIUS module.

`WebCal`
If this Service is disabled, the Account user cannot use the Calendaring functions of the WebUser and XIMSS Interfaces, and the user cannot be authenticated using the CalDAV protocol.

`WebSite`
If this Service is disabled, HTTP and WebDAV access to this Account is disabled.

`PBX`
If this Service is disabled, the Account cannot use the PBX services.

`HTTP`
If this Service is disabled, sessions and real-time applications created for this Account cannot use outgoing HTTP transactions.

The Server checks the Account and the Account Domain settings. Only if the Service is enabled for both the Account and the Account Domain, that service can be used with this Account. See the Domains Settings section for more details.

If you select the `default` option, the Enabled Services for this Account are defined using Domain Default Account Settings or the Server-wide/Cluster-wide Default Account Settings.

**Note:** please note a difference between the Default Account settings and the Enabled Services specified for the Domain: while you can override the Default Account Settings for some Account by explicitly specifying the enabled Services for that Account, you cannot override the Enabled Services specified for the Domain. If the Default Account Settings disable POP and IMAP access, you can explicitly enable POP and IMAP access for a particular account. But if POP and IMAP access is disabled with the Domain Settings, no Account in that Domain can be accessed via these protocols.

## Access Settings

| Access | | | |
|---|---|---|---|
| | POP: default(3) | IMAP: | default(10) |
| Session Limits: | AirSync: default(10) | XMPP: | default(10) |
| | XIMSS: default(10) | WebUser: | default(10) |
| | POP Login Limit: 3　　in　10 min | | |

`Session Limits`

These settings limit the number of concurrent sessions accessing this Account. If the number of the already opened sessions exceeds the specified limit, a new login operation is rejected with an error code being sent to the client application.

`POP Login Limit`

This setting limits the frequency of POP Logins into this Account. If the number of POP logins exceeds the specified limit, the session is rejected with an error code being sent to the mail client. The user should change the POP client settings to make it check mail less often.

## Mail Storage Settings

Open the Mail Settings page in the Mail section of the Account Settings:

| Mailbox Storage | | | |
|---|---|---|---|
| Mail Storage Limit: | default(unlimited) | Used: | |
| Mailbox Limit: | default(unlimited) | Used: 31 | |
| Archive Messages after: | 365 day(s) | Delete Messages after: | default(Never) |
| New Mailbox Format: | default(Text) | Non-Mail Folders visible via IMAP: | default(No) |
| Erase Deleted Messages: | Yes | Encrypted Mailbox Creation: | default(Prohibit) |

`Mail Storage Limit`

This option is used to specify the maximum total size of the all Account Mailboxes. If a new incoming message cannot be stored in an Account, because the Account size would exceed the specified limit, the message is rejected and the message sender receives an error report.

The current Mail Storage usage value is shown as a button, if the page is viewed by a System Administrator. By clicking this button you can re-calculate the usage storage counter if that counter was de-synchronized.

`Mailbox Limit`

This option is used to specify the maximum number of Mailboxes that can be created in this Account.

`Archive Messages after`, `Delete Messages after`

See the [Chronos](#) section for the details.

`New Mailbox Format`

This setting is displayed for multi-mailbox Accounts only. It specifies the default [format](#) for all new Mailboxes created in this Account.

`Non-Mail Folders visible via IMAP`

This setting controls if IMAP clients see non-Mail (Calendar, Contacts, etc.) Mailboxes. See the [IMAP](#) module section for more details.

`Erase Deleted Messages`

When this option is selected, the messages being deleted from the Account Mailboxes are rewritten with garbage first.

`Encrypted Mailbox Creation`

This setting specifies if the Account user is allowed to create Encrypted Mailboxes.

## Incoming Mail Transfer Settings

| Incoming Mail Transfer | | | |
|---|---|---|---|
| Incoming Mail Limit: | 30    in    10 min | Incoming Message Size Limit: | 30M |
| Delay New Mail if: | default(100)    % full | Send Alerts if: | default(80)    % full |
| Send Notice if: | default(90)    % full | | |
| Allowed Mail Rules: | default(Filters only) | Used: 7 | |
| RPOP modifications: | default(Prohibit) | Used: 3 | |
| Accepts Mail to "all": | default(Yes) | | |

`Incoming Message Size Limit`

This option is used to specify the maximum size of an E-mail Message that can be delivered to the Account.

`Incoming Mail Limit`

This option is used to limit the number of E-mail Messages an Account can receive over the specified period of time.
See the Local Delivery Module section for more details.


Delay New Mail

If the Account mail storage size is limited, and the specified percent of that limit is already used, or it would be used when the new message is added, message delivery to this Account is suspended. The Local Delivery module settings specify what actually happens to the Account message queue in this case.

Send Alerts

This option specifies when the Storage Quota Alerts should be sent to the Account user.
The Alert message text is a Server String and it can be customized.

Send Notice

This option specifies when the Local Delivery module should compose and store an "over quota" message in the Account INBOX. If this Notice Message is stored, no new Notice Message will be composed and stored for the next 24 hours.
The Notice Message Subject and the Message text are Server Strings and they can be customized. There are two different Notice Message bodies - one is used when an incoming message has been delivered, and the other one - when an incoming message is too big to be delivered to the Account.
**Note:** the Notice Messages are not submitted to the Queue, they are composed with the Local Delivery module and they are stored directly in the Account INBOX.

To disable these 3 options, set their values to 101%.


Allowed Mail Rules

This setting tells the Server if the user is allowed to specify automated Rules that instruct the Server how to process incoming E-mail messages.

No

If this option is selected, only the administrator can specify the automated rules for this user.

Filter Only

If this option is selected, the user can specify only the following actions:
Discard, Reject, Stop Processing, Mark, Add Header, Tag Subject, Store in, Copy Attachments to, and Store Encrypted in.

All But Exec

If this option is selected, the user can specify any action, but the Execute action.

Any

If this option is selected, the user can specify any action.


Click the Mail Rules link to specify the rules to be applied to all incoming E-mail messages directed to this Account.

If an administrator creates an Automated Rule containing actions the Account user is not allowed to specify, the user will be able to view that Rule, but not to modify any part of it.


RPOP Modifications

This setting tells the Server if the user is allowed to specify remote host (RPOP) accounts that the RPOP module should poll on the user's behalf.
If this option is disabled, only the administrator can specify the RPOP accounts for this user.
Click the RPOP link to specify the remote accounts to be polled on behalf of this user.

Accept Mail to all

This setting tells the Server to store messages directed to the all@domain address in the Account INBOX.


## Outgoing Mail Transfer Settings

| Outgoing Mail Transfer | | | | | |
|---|---|---|---|---|---|
| Outgoing Mail Limit: | 15 | in | 10 min | Outgoing Message Size Limit: | 30M |
| Outgoing Recipients Limit: | 20 | in | 10 min | Max Recipients per Message: | 15 |
| 'From' Address Restrictions: | Relaxed | | | 'From' Name Restrictions: | default(None) |

| Add Trailer to Sent Mail: | default(Yes) |
|---|---|
| Reroute External Recipients to: | * |

**Outgoing Mail Limit**

This option is used to limit the number of E-mail Messages composed and submitted on behalf of this Account, over the specified period of time.

If the number of messages submitted during the specified period of time exceeds the specified limit, the Account user's ability to submit messages is suspended (for the specified period of time).

**Outgoing Message Size Limit**

This option is used to specify the maximum size of an E-mail Message that can be composed and submitted by this Account.

**Outgoing Recipients Limit**

This option is used to limit the number of recipients to whom E-mail Messages can be submitted on behalf of this Account, over the specified period of time.

If the number of recipients submitted during the specified period of time exceeds the limit, the user's ability to submit messages is suspended.

**Max Recipients per Message**

This option is used to specify the maximum number of recipients in a single E-mail Message that can be submitted by this Account.

**'From' Address Restrictions**

This option is used to specify what e-mail address can be used in `From:` header in messages submitted by this Account.

**None**

If this option is selected, any address can be used, including a non-existent one.

**Strict**

If this option is selected, only the Account name can be used as the address.

**Relaxed**

If this option is selected, the following can be used:

- any address which is routed to the Account through Alias, Forwarder, or Router entry.
- a Group name, if the Account is a member of that Group.
- another Account name from the same CommuniGate server, if this Account has `CanImpersonate` Domain access right, or if that Account had granted this Account the `Delegate` Account access right.

**'From' Name Restrictions**

This option is used to specify what title can be used in `From:` header in messages submitted by this Account.

**None**

If this option is selected, any name can be used.

**Strict**

If this option is selected, the following can be used:

- the exact value of Real Name attribute of this Account, if the 'From' address is routed to the Account.
- the exact value of Real Name of a Group, if the 'From' address is routed to that Group and this Account is the member of the Group.
- the exact value of Real Name attribute of another Account, if the 'From' address is routed to that Account, and if this Account has `CanImpersonate` Domain access right, or if that Account had granted this Account the `Delegate` Account access right.

**Relaxed**

If this option is selected, the following can be used:

- an empty string.
- any string containing the value of Real Name attribute of this Account, if the 'From' address is routed to the Account.
- any string containing the value of Real Name of a Group, if the 'From' address is routed to that Group and this Account is the member of the Group.

- any string containing the value of [Real Name](#) attribute of another Account, if the 'From' address is routed to that Account, and if this Account has `CanImpersonate` [Domain access right](#), or if that Account had granted this Account the `Delegate` [Account access right](#).

These options are used to prevent system abuse by the system own users, and for mitigation of damage from compromised Accounts used by hackers.

These limits and restrictions are applied to the messages submitted from *authenticated sources*: via SMTP using the AUTH operation, via the WebUser Interface and XIMSS clients, via the MAPI module, via the [POP](#) module XTND XMIT command, via [AirSync](#) clients, etc.

**Note:** some of users can send messages via SMTP without using the AUTH operation, because they send from the network addresses specified in the [Client IP Addresses](#) list, or because they use the [Read-then-Send](#) method. In this case the messages they submit cannot be attributed to any Account and those messages are not counted, and their From: headers are not checked.

If you want to apply the Outgoing Flow Control settings to all messages submitted by your users, you should force all of them to use the SMTP AUTH operation by enabling the Force SMTP AUTH option in the Domain Settings.

**Note:** the messages generated by Account-level or Domain-wide Rule action, including ones generated by the [Copy All Mail To](#) simplified Rule, are also fall under the restrictions. So when the restrictions are on, the user may be able to use only the "Send on behalf of this Account" (`Forward to`) action, but not `Redirect to` and `Mirror to` actions which preserve the original `From:` header.

`Add Mail Trailer`
    This setting tells the Server to append the trailer text (specified in the [Domain Settings](#)) to all messages this user composes using the [WebUser Interface](#).

`Reroute External Recipients to`
    This setting specifies how the addresses of the external message recipients are changed.
    If the setting value is `*` or empty string then the recipients are unchanged. Otherwise the `*` symbol is replaced with the original recipient address and the [Router](#) restarts trying to route this new address.

    Sample:
        The Domain `Reroute External Recipients to` option is set to
        `*@smarthost.com._via`
        The outgoing messages from users of this domain will be sent to intended recipients through the `smarthost.com` server.

    Sample:
        The Account `Reroute External Recipients to` option is set to
        `error`
        The user will be able to send only locally within the Server, attempts to send to outside will cause an error.

    **Note:** This rerouting is applied only to external addresses (directed to SMTP Module), not applied to local addresses.

Any of these Settings can be set to the `default` value, in this case the setting value is taken from the Domain Default Account Settings or the server-wide/cluster-wide Default Account Settings.

## Signaling Settings

Open the Call Settings page in the Real-Time section of the Account Settings:

| Calls | | |
|---|---|---|
| Allowed Call Rules: | Any | Used: 10 |
| Concurrent Calls: | 2 | Used: 0 |
| Incoming Calls Limit: | unlimited    in   60 min | Call Logs:   default(Enabled) |
| Outgoing Calls Limit: | unlimited    in   60 min | Call Info:   default(Enabled) |

`Allowed Call Rules`
    This setting tells the Server if the user is allowed to specify automated Rules that instruct the Server how to process incoming [Signals](#).

    `No`
        If this option is selected, only the administrator can specify the automated Signal Rules for this user.

Any
>	If this option is selected, the user can specify any action.

Click the Call Rules link to specify the rules to be applied to all incoming Signals (calls) directed to this Account.

If an administrator creates an Automated Rule containing actions the Account user is not allowed to specify, the user will be able to view that Rule, but not to modify any part of it.

Concurrent Calls
>	This setting specifies the maximum number of concurrent calls this Account can be involved in (as a caller or a callee).
>	**Note:** if the Call Info option is disabled, the number of concurrent calls cannot be defined and it cannot be limited.

Call Logs
>	If this option is enabled, the Signal component places call log records into the `private/logs/` files in the Account File Storage.

Call Info
>	If this option is enabled, the Signal component stores the current calls information in the Account data, from where it can be retrieved using the dialog Signal package, and other methods.
>	Disable this option if the Account is a service one, handling many concurrent calls, and there is no need to retrieve the current call information.

Incoming Calls Limit
>	This setting specifies the maximum number of incoming calls this Account can receive over the specified period of time.

Outgoing Calls Limit
>	This setting specifies the maximum number of outgoing calls this Account can make over the specified period of time.

| Device Registration | | |
|---|---|---|
| Registered Devices: | default(10) | Used: 2 |
| RSIP modifications: | default(Enabled) | Used: 1 |

Registered Devices
>	This setting specifies the maximum number of "Contacts" (devices) the Server can register for this Account. The Used field shows the number of the currently registered devices.

RSIP modifications
>	This setting specifies if the Account user is allowed to modify the Account RSIP settings. The Used field shows the current number of the Account RSIP records.

| Instant Messaging and Presence | | | | |
|---|---|---|---|---|
| Roster Limit: | default(20) | | Used: 7 | |
| Send IMs to SIP Devices: | If Supported | | IM Logs: | default(Enabled) |
| Outgoing NOTIFY Requests Limit: | 100 | in 15 sec | Map Call Dialog Status to Presence: | default(Disabled) |

Roster Limit
>	This setting specifies the maximum number of Roster elements ("Buddies") for this Account.

Send IMs to SIP Devices
>	If this option is set to Enabled, incoming Instant Messages are delivered to all registered SIP devices;
>	if this option is set to If Supported, incoming Instant Messages are delivered only to those registered SIP devices that declared support for the MESSAGE SIP Method;
>	if this option is set to Disabled, incoming Instant Messages are not delivered to registered SIP devices.

IM Logs
>	If this option is enabled, incoming and outgoing Instant Messages are stored in the Account File Storage.

Outgoing NOTIFY Requests Limit
>	This setting specifies the maximum number of NOTIFY requests (such as presence updates, call dialog status updates, etc.) this Account can send over the specified period of time. Increase this setting as you increase the allowed Roster Limit: when the Account changes its presence state, a NOTIFY request with the new presence info is sent to each Roster item (to each "buddy").

Map Call Dialog Status to Presence
>	If this option is enabled, and this Account has active calls, then the Account presence state is set to "on-phone".

The Instant Messages processing is also controlled with [Account Preferences](#).

# File Storage Settings

Open the File Settings page in the Files section of the Account Settings:

| File Storage | | |
|---|---|---|
| File Storage Limit: | 100M | Used: 1340K |
| Files Limit: | default(1000) | Used: 33 |
| File Size Limit: | default(unlimited) | |
| Add Banner to HTML: | default(Yes) | Default file for HTTP: `default.html` |

File Storage Limit
> This option is used to specify the maximum total size of the all files in the Account [File Storage](#). If this option is set to zero, the Account File Storage is disabled.

File Size Limit
> This option is used to specify the maximum size of a file that can be stored in the Account File Storage.

Files Limit
> This option is used to specify the maximum number of all files in the Account [File Storage](#).

Add Banner to HTML
> This setting tells the Server to insert the Web banner code (specified in the [Domain Settings](#)) to all HTML files retrieved from this Account [File Storage](#).

Default Web Page
> When an HTTP URL for a File Storage file does not specify a file name (`http://domain:port/~account/` or `http://domain:port/~account/subDir/`), a file with the Default Web Page name is retrieved.

# WebUser Interface Settings

Open the Account Settings page to modify the WebUser Interface settings:

| WebUser Interface | | |
|---|---|---|
| Hidden Skins: | | Hidden Pronto! Modules: `Twitter,Video` |
| Banner Parameters: | | |

Hidden Skins
> This option is used to specify the names of the WebUser Interface (and Pronto!) [Skins](#) that should be hidden from this Account user. If several names are specified, separate them with the comma (`,`) symbol. To hide the "unnamed" Skin, specify the `unnamed` name.

Hidden Pronto! Modules
> This option is used to specify the names of the Pronto! modules that should not be loaded for this Account user. If several names are specified, separate them with the comma (`,`) symbol.

Banner Parameters
> This setting value is passed to the [External Banner System](#) when it is requested to generate an advertising banner for this Account session.

---

# Account Aliases

Each Account can have Aliases (alternative names).

If the `john_smith` Account has the `jsmith` and `j.smith` Aliases, E-mail directed to jsmith and to j.smith will be stored in the `john_smith` Account, and the Signals directed to to jsmith and to j.smith will be delivered to the `john_smith` Account.
To access the john_smith Account via POP, IMAP, XMPP, XIMSS, WebUser, or any other client application both the user names `jsmith`

and `j.smith` can be specified in the client application settings.

| Aliases |
|---|
| |

You can modify existing Aliases, you can add an Alias by typing a new name in the empty field, and you can remove an Alias by deleting it from its field. Use the Update button to update the Account Aliases list.

Alias names should not be the same as the name of some other Account, or other Object in the same Domain.
The Alias names should meet the [Domain Object](#) name restrictions.

You can specify several Aliases in one field by separating them with the comma (`,`) symbol.

# Account Telephone Numbers

Each Account can have zero, one, or more Telephone (PSTN) numbers assigned to it.

The Server maintains a global list of all Telephone Numbers assigned to all Accounts in all Domains.

Telephone numbers should be specified in the E.164 format: `+country_code area_code local_number`. The number should contain only digits and it can start with the plus (`+`) symbol.

**Note:**

- Only when a [Signal](#) (a call) comes to your CommuniGate Pro Server or Cluster, these Telephone Number mappings take effect.
- An assigned Telephone Number should be registered with one of the PSTN Gateways. When a PSTN call is made to that number, the Gateway should receive the call and it should direct the call to your CommuniGate Pro Server via a VoIP protocol (such as [SIP](#)). When a call made to a PSTN number arrives to the Server, it is usually still directed to the dialed PSTN number, and not to the user Account name. The Server uses its global list of assigned Telephone Numbers to route the call to the proper Account.
- Users may want to register their assigned Telephone Numbers with one of the global [ENUM services](#).
  If a Telephone number is linked to a CommuniGate Pro Domain using such a service, VoIP calls made by users of all VoIP systems employing that ENUM service will be routed to the CommuniGate Pro Account directly, via the Internet, bypassing PSTN.

To manage the Account Telephone Numbers, open the WebAdmin Account Management pages, then open the Call Settings page in the Real-Time section:

| Telephone Numbers |
|---|
| |

You can modify existing Telephone Numbers, you can add a Telephone Number by typing a new name in the empty field, and you can remove a Telephone Number by deleting it from its field. Use the Update button to update the Telephone Numbers list.

See the [PSTN](#) section for more details.

# Access Rights

The CommuniGate Pro Server maintains an Access Control List (ACL) for every Account.

The Access Control Lists are used to control what other users can do with this Account.

A Server Administrator with the [All Domains](#) access right has all access rights to all Server or Cluster Accounts.

The Account owner can grant certain limited access rights to other users, controlled with a [Access Control List](#).

The following Account access rights are supported:

c (CreateMailbox)
    If you grant a user the CreateMailbox access right, that user will be able to create Mailboxes on the "top" of your Account (i.e. Mailboxes that are not sub-Mailboxes of existing Mailboxes in your Account).

x (Delegate)
    If you grant a user the Delegate access right, that user will be able to subscribe to the Account Event packages.

j (CallControl)
    If you grant a user the CallControl access right, that user will be able to access the information about your real-time communications. This information includes the data about Signal objects handling your incoming calls, and it can be used to intercept your incoming calls ("call pick-up").

The Account Access Control Lists can be set and modified using the WebUser Interface, a XIMSS, or a MAPI client.

# Renaming Accounts

If you want to rename an Account, open its Settings page, and enter a new Account name into the New Account Name field. Click the Rename Account button.

If there is no other Object with the same name as the specified new Account name, the Account is renamed and its Account Settings page should reappear on the screen under the new name.

You cannot rename an Account when it is in use.

> New Account Name:

If you are a Domain Administrator, you should have the Can Create Accounts Access Right to rename Accounts in your Domain.

You can move an Account into a different Domain, if you specify the new Account name as *newName@domainName*.
If you are a Domain Administrator, you should have the Can Create Accounts Access Right for both Domains.

# Removing Accounts

If you want to remove an Account, open its Settings page, and click the Remove Account button. The confirmation page should appear.

If you confirm the action, the selected Account, all its Mailboxes, Settings, and other Account-related data files will be permanently removed from the Server disks.

The Account Aliases and all Mailing List owned by this Account will be removed, too.

You cannot remove an Account when it is in use.

If you are a Domain Administrator, you should have the Can Create Accounts Access Right to remove Accounts from your Domain.

# Default Account Settings

An Account setting can have the default value. In this case the actual setting value is taken from the Default Account Settings for the Account Domain.
You can modify the Default Account Setting values by clicking the Account Defaults link on any Domain administration page of the WebAdmin Interface.

The Default Account Settings page resembles a regular Account Settings page.

The Domain Default Account Settings themselves can be assigned the default value.

In this case the setting value is retrieved from the Server-wide or Cluster-wide *Default Account Settings*.
You can modify the server-wide Default Account Settings by clicking the Account Defaults link on the Domains (Domain List) page.

A Dynamic [Cluster](#) installation maintains separate server-wide Default Account Settings for all Accounts in non-Shared (Local) Domains, and cluster-wide Default Account Settings for all Accounts in the Shared Domains. In the Cluster environment, the Default Account Settings page displays links that allow you to switch between the Server-wide and Cluster-wide Default Settings.
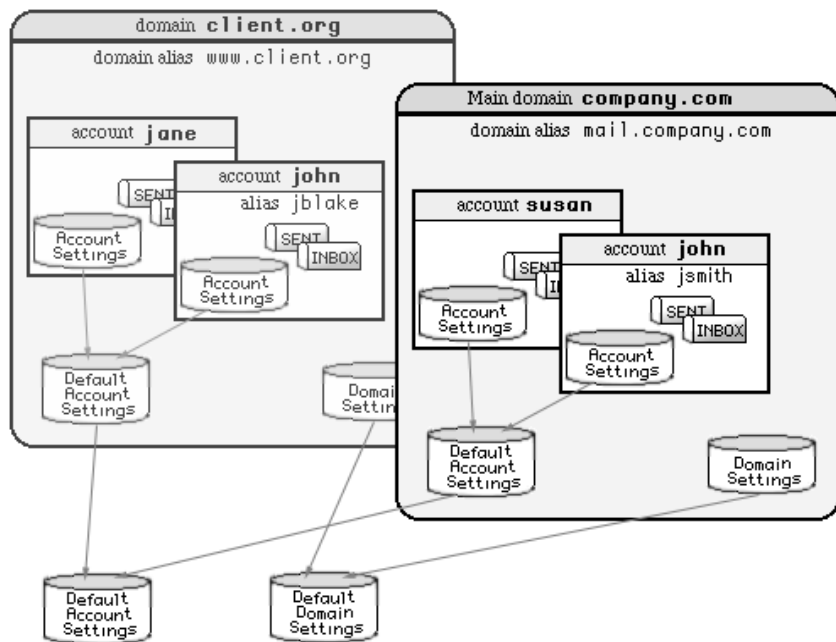


Example:

| | | |
|---|---|---|
| The global (Server) | Default Account Settings: | `Storage Limit = 10Mbytes` |
| The `company.dom` | Default Account Settings: | `Storage Limit = 30Mbytes` |
| The `client1.dom` | Default Account Settings: | `Storage Limit = default` |

Now:

- If you create an Account in any Domain, and set its `Storage Limit` to some value, that value will be used.
- If you create an Account in the `company.dom` Domain, and set its `Storage Limit` value to `default`, the Account will be able to keep up to 30Mbytes of mail (the Default Account Setting for that Domain).
- If you create an Account in the `client1.dom` Domain, and set its `Storage Limit` value to `default`, the Account will be able to keep up to 10Mbytes of mail (the global Default Account Setting for the Server).

When you serve many Accounts, you should try to specify most of the setting values as `default`, so you can easily change those settings for all Accounts. If some Account should be treated differently, you should explicitly specify the required setting value for that Account.

---

# Class of Service

A Class of Service is an additional named Default Account Settings set. You can have several sets within each Domain, and you can specify a set to use for each Account, so Accounts in the same Domain can have different Default Settings.

To create a new Class of Service, use the WebAdmin Interface to open the Account Defaults page (a Domain one, a Server-wide, or a Cluster-wide one).

If you are a Server Administrator, or a Domain Administrators with the [ServiceClasses Access Right](#), you can create additional Classes of Service:

New Class of Service Name:

Enter the new Class of Service name and click the Create Class of Service button. A new Class of Service will be created.

The Account Defaults page lists all created Classes of Service:

| Classes of Service | |
|---|---|
| FreeUser | Basic |
| Premium | Business |

To open the Class of Service settings, click its name. The Class of Service settings page is the same as the Account Defaults page.

Account Defaults can be viewed as an unnamed Class of Service.

If you are a Server Administrator, or a Domain Administrators with the ServiceClasses Access Right, you can rename or remove Classes of Service. Use the WebAdmin Interface to open the Class of Service page, and scroll to the bottom of the page:

New Class of Service Name:

When there is at least one Class of Service created on the Domain or Server/Cluster level, the Account Settings page includes the Class of Service setting. If you are a Server Administrator, or a Domain Administrators with the ServiceClass Access Right, you can modify this setting:

Class of Service:    Premium

When an Account Class of Service setting is not empty, the following algorithm is used to retrieve an Account settings:

- If the Account has a setting value explicitly assigned, that value is used.
- If the Domain has a Class of Service with the specified name, and that Class of Service settings contain a setting value, that value is used.
- If the Server-wide (a Cluster-wide for Accounts in shared Domains) has a Class of Service with the specified name, and that Class of Service settings contain a setting value, that value is used.
- Finally, the Server-wide (a Cluster-wide for Accounts in shared Domains) Account Default settings are used to retrieve the setting value.

You can create a Domain Class of Service with the same name as a Server-wide (or Cluster-wide) Class of Service to override some of its settings.

---

# Account Template

When you need to create many Accounts, you may want to specify some non-default setting for all new Accounts. Each Domain has its own Account Template, and you can modify it by clicking the Template link on the Account List page.

The Accounts Template page resembles a regular Account Settings page.

All the settings set there will be copied to all newly created Accounts in this Domain.

**Note:** The Default Account Settings and Account Template are quite different. The Account Template is used only when an Account is being created. All template settings with non-default values are copied to the new Account settings. If you modify the template settings after an Account has been created, those Account settings will not change.

Besides the initial, non-Default setting values, the Account Template can be used to instruct the Server to create additional Mailboxes in each new Account (by default only the INBOX Mailbox is created), to subscribe the Account to certain Mailboxes, and to create Mailbox Aliases in all newly created Accounts.

| Additional Mailboxes | Locked |
|---|---|
| IPF.Appointment | |

Enter a name into the empty field to add a Mailbox name to the list.

For non-mail Mailboxes, specify the Mailbox Class from the pop-up menu.

If you select the [Lock](#) checkbox, it will be impossible to delete or rename the created Mailbox.

In this sample, when a new multi-mailbox Account is created in this Domain, the mail Mailboxes `Sent` and `Drafts`, and the calendar Mailbox `Calendar` will be created in that Account, along with the `INBOX` Mailbox.
The Account users will not be able to delete or rename the Calendar Mailbox.

| Initial Subscription |
|---|
|  |

See the [Mailboxes](#) section to learn about Mailbox Subscriptions.

Creating initial non-empty subscription:

- simplifies the initial set-up of some client mailers that can access only those Account Mailboxes that are included into the Mailbox Subscription list;
- helps new users to subscribe to public Mailboxes containing administrative information, news, etc.

| Initial Mailbox Aliases | |
|---|---|
| **Alias Name** | **Mailbox Name** |
|  |  |

See the [Mailboxes](#) section to learn about Mailbox Aliases.

Specifying a non-empty list of Mailbox Aliases simplifies the initial set-up for Microsoft Outlook users that need access to public Mailbox and other [Foreign Mailboxes](#), but cannot use their mailers to access foreign Mailboxes directly.

| Greeting E-mail |
|---|
|  |

This field can contain a mail message in the RFC822 format. If this field is not empty, then the specified message is stored in the INBOX Mailbox of every newly created Account.

The text can contain the following macro combinations, replaced with the newly created Account data:

- `^A` - the newly created Account name.
- `^D` - the Domain name.
- `^E` - the newly created Account Real Name.

The `Date:` header field is automatically added to the stored messages.

The message text can start with a `[charsetName]` prefix, then the text will be converted from UTF-8 to the specified character set. Specify the Content-Type header field with the proper the `charset=` parameter:

| Greeting E-mail |
|---|
|  |

Templates can be used to generate an initial default Web (HTML) page in the File Storage for all newly created Accounts:

| Initial Personal Home Page |
| --- |
| |

This field can contain an HTML text. If this field is not empty, then the specified text is stored as the Default Web Page file in the File Storage of each newly created Account.

# Importing User Account Information

The built-in Account Loader allows the administrator to register sets of users. The user names and Account attributes should be placed into a tab-delimited text file on the administrator (client) computer, and that file should be uploaded to the server using the Import field.

Click the browse button to select a file on your local system, and then click the Import Accounts button to create Accounts listed in the selected file.

| | **Template** |
| --- | --- |
| Multi-Mailbox | |
| no file selected | |

Below is a sample IMPORT file:

```
Name        Type                Ignore          Storage         Aliases
johnd       MultiMailbox        sales dept      50M
susan       MultiMailbox        mgmnt           10M             susan.s,susan_smith
sales       MultiMailbox        dummy           30M
info        MultiMailbox        dummy           50M             help
```

**Note:** The import file must be prepared on the client computer (on the computer you use to run your browser). The browser allows you to upload files from disks connected to that computer, not to the CommuniGate Pro Server computer.

**Note:** When using Netscape and some other Unix browsers, make sure that the file name ends with the `.txt` suffix - otherwise the browser won't upload the file as a text one, and the file will be ignored.

**Note:** The MacOS 9.x versions of the Microsoft Internet Explorer upload Macintosh files in the encoded `x-macbinary` format if the file contains a *resource fork*. Most text files created with Macintosh text editor applications contain resource forks that keep the information about the file fonts, file window position, and other Macintosh data. Such files cannot be used as import files with the Microsoft Internet Explorer browser. Either use a text editor application that saves text files without resource forks or use a browser that uploads Macintosh files without encoding.

The first file line describes the file contents. It should contain tab-delimited names of Account attributes. The following names are supported:

`Name`
> This column contains the Account names. This attribute is not required to be in the first column, but it must exist. All other attributes are optional.

`RealName`
> This column contains the Account user "real name".

Type
       This column contains the Account type (`MultiMailbox`, `Text Mailbox`, etc.). If the file does not contain this column, or this field is empty, the Account type selected on the Account List WebAdmin page is used.

Password
       This column contains the Account password. If the file does not contain this column, or this field is empty, the CommuniGate Password and the Use CommuniGate Password settings are taken from the Domain Account Template.

UnixPassword
       This column can be used instead of the `Password` column. If it exists, it should contain `crypt`-encrypted Account passwords. The Account Loader will add the binary prefix to those strings, so these CommuniGate passwords will be used as `U-crpt` encrypted passwords. See the [Migration](#) section for more details.

Storage
       This column contains the maximum Account Mail Storage size (in bytes, or in kilobytes, if the number is followed with `K`, or in megabytes, if the number is followed with `M`). The column data can contain `-1` or `unlimited` to specify unlimited storage.

Aliases
       This column contains the Account Aliases; to specify several Aliases, separate them with the comma symbol.

Telnums
       This column contains the Account Telnums; to specify several Telnums, separate them with the comma symbol.

MailInRules
       This column contains the Account [Mail Processing Rules](#). Rules should be represented in the internal format, as an [array](#) of individual Rules. Each Rule is an array, where the first element is the Rule priority, the second element is the Rule Name [string](#), the third element is the Rule conditions array, and the last element is the Rule Actions array.

SignalInRules
       This column contains the Account [Signal Processing Rules](#).

Ignore
       This column is ignored. An Account list file can contain several Ignore columns.

*setting name*
       You can use columns that contain initial values for various additional Account settings (File Storage file and size limit, type or Rule actions enabled, etc.). Any additional column should have the same name as the selected Account setting name (keyword). For example, you can use the column named `MaxWebSize` to specify the storage limit for the Account File Storage, and you can also use the column named `MaxAccountSize` instead of the `Storage` column.

*Custom Setting*
       You can use columns that contain initial values for various [Custom Account Settings](#). For example, if the Directory Integration page contains the Custom Setting `city`, you can include a column named `city` in your Account Import file.

If the first line is parsed, all other lines are processed. Each line should contain tab-delimited fields, with the field contents specified in the first line. A line can contain less fields than the first line, in this case missing fields are processed as empty fields.

Attribute values for empty and missing fields are taken from the [Account Template](#).

If an error occurs while processing some file line (missing name field, duplicate name, etc.), all Accounts created while processing previous lines are removed, and the number of the line that caused the problem is displayed. You can fix the file and try again.

| Objects | Domains | Accounts | Groups | Forwarders | Named Tasks | Chronos |
|---------|---------|----------|--------|------------|-------------|---------|

# Groups

- **Creating a New Group**
- **Specifying Group Settings**
- **Group Member Processing**
- **Renaming Groups**
- **Removing Groups**

CommuniGate Pro Domains can contain Group objects. A Group contains a set of group members. Group members are other objects in the same Domain and/or external E-mail addresses. Each Group has its own Settings.

Messages sent to a Group are processed with the LIST module. The module copies the messages without any modification of any header field, and it resubmits them using the Group members to fill the envelope recipient list.

Administrators can create Group objects to simplify mailing to logical groups of users ("marketing", "sales", etc.). A message can be addresses to a simple `groupname@domain.dom` address, and the server will distribute it to all groupname Group members.

Real-time (IM, audio, video, etc.) requests sent to a Group are processed with the Signal component. These requests are "forked" to all Group members.

# Creating a New Group

Groups in any Domain can be created by the Server Administrator if the administrator Account has the Can Modify All Accounts And Domain Settings access right.

A Domain Administrator can create, remove, and rename groups only if the CanCreateGroups access right is granted to the administrator account.

To create a new group, type a new group name into the field on the right side of the Create Group button. The selected Group name should meet the Domain Object name restrictions.

Click the Create Group button. When a new group is created, its name appears in the list. The Server automatically displays the Settings page for the new group.

# Specifying Group Settings

To specify Group Settings, click the Group name on the Objects list page. The Group Settings page appears.

| Real Name: | |
|---|---|
| Report Delivery to Group | Set Reply-To to Group |
| Expand Member Groups | Reject Automatic Messages |
| Remove Author from Distribution | Remove To and Cc from Distribution |
| Disable E-mails | Disable Signals |

| **Members** |
|---|
| |

Members

> This is the list of all Group members. If the member name does not contain a domain part, it specifies an object in the same Domain: a Domain Account or some other Group.
> The last empty element of the table allows the administrator to add a new member to the Group. To delete a member from the Group, delete the member name and click the Update button.
> You can enter several addresses in one field, separating them with the comma ( , ) symbol. After you click the Update button, each address will be displayed in a separate field.

RealName

> A brief description of the Group. This string is used to compose the comment for this Group E-mail or Signal address.

Report Delivery to Group

> If this option is selected, then a delivery report (if requested) is generated as soon as an E-mail message is copied and re-submitted for delivery to all Group members. If subsequent delivery to any Group member fails, error reports are not generated.
> If this option is not selected, delivery to this Group is processed as "relaying", and the delivery notification options are copied to addresses of all Group members.
> If delivery to any Group member fails, the sender gets an error message.
> If a message was sent with delivery notification requested, the sender will get notification delivery from all Group members.

Set Reply-To to Group

> If this option is selected, the Reply-to: header pointing to the Group address is added to the message copy before it is sent to Group members. This ensures that replies to a message sent to this Group will go back to the Group, not to the message author.

Expand Member Groups

> If this option is selected, the Group members are checked before a message is copied and sent to member addresses. If a Group member is some other Group in the same Domain, then that Group members are extracted and inserted into the address list. If that Group also has this option enabled, the extracted members are checked, too. This option allows to process Group delivery more efficiently (only one message copy is created for all recipients) and it also helps to avoid duplicates and mail loops.
> If the Group contains 2 other groups (sub-groups) as members and those sub-groups contain the same address, then only one copy of the message is delivered to that address if the Expand option is enabled. If this option is

disabled, the copy of the original message will be delivered to both sub-groups, and each sub-group will send its copy of the original message to that address.

Reject Automatic Messages
> If this option is selected, automatic messages (i.e. not "Human-Generated" messages) cannot be sent to this Group.

Remove Author from Distribution
> If this option is selected, the message From: address is removed from the (optionally expanded) members list.

Remove To and Cc from Distribution
> If this option is selected, all addresses from the message To and Cc fields are removed from the (optionally expanded) members list.

Disable E-mails
> If this option is selected, E-mail messages sent to this Group are rejected.

Disable Signals
> If this option is selected, Real-Time Signals sent to this Group are rejected.

---

# Group Members Processing

The CommuniGate Pro Server can do certain "reverse processing" operations with Group members by finding Groups containing a certain Domain object (such as an Account, a Group, or a Forwarder).

When a Domain object is removed, the Server removes the object name from all Groups in that Domain.

When a Domain object is renamed, the Server updates the object name in all Groups in that Domain.

When an Account user is trying to open a Foreign Mailbox, the Server checks the Mailbox Access Rights granted to all Groups containing that Account.

**Note:** to allow the Server to find an object in Group member lists, the member lists should contain the "real" object name, not its Alias (or any other name routed to the object name).

---

# Renaming Groups

If you want to rename a Group, open its Settings page, fill the New Group Name field, and click the Rename Group button.

If there is no other Object with the same name as the specified new Group name, the Group is renamed and its Group Settings page should reappear on the screen under the new name.

New Group Name:

---

# Removing Groups

If you want to remove a Group, open its Settings page, and click the Remove Group button.

**CommuniGate Pro**

Objects   Domains   Accounts   Groups   **Forwarders**   Named Tasks   Chronos

# Forwarders

- **Creating Forwarders**
- **Updating Forwarding Address**
- **Renaming Forwarders**
- **Removing Forwarders**

CommuniGate Pro Domains can contain Forwarder objects. A Forwarder is an E-mail address associated with the Forwarder name. All E-mail messages and Signals directed to the Forwarder name are rerouted to the Forwarder address.

Forwarders work exactly as the Router alias records. A Forwarder `jim` in the `client1.com` Domain containing the `john@otherdomain.dom` address acts as the following Router record:

        Relay:<jim@client1.com> = john@otherdomain.com

Forwarders allow the Server administrator to keep the Router table small.

Forwarders can be set by Domain Administrators, while the Router can be modified by the Server Administrator only.

A Server Administrator with the Can Modify All Accounts And Domain Settings access right can create, update, rename, and remove Forwarders in any Domain.

Domain Administrators can create, update, rename, and remove Forwarders in their Domains if they have the CanCreateForwarders access right.

## Creating Forwarders

To create a Forwarder, open the Domain Object list page. The list of the Domain Objects (optionally including the Forwarders) appears:

| Forward to: |
| --- |

To create a new Forwarder enter its name and the address to forward to, then click the Create Forwarder button. The selected Forwarder name should meet the Domain Object name restrictions.

## Updating Forwarding Address

To update a Forwarder, click the Forwarder name on the Objects list page. The Forwarder Settings page appears.

| Forward to: |
| --- |

Update the string in the Forward to field and click the Update button to modify the forwarding address.

# Renaming Forwarders

If you want to rename a Forwarder, open its Settings page, fill the New Forwarder Name field, and click the Rename Forwarder button.

If there is no other Object with the same name as the specified new Forwarder name, the Forwarder is renamed and its Settings page should reappear on the screen under the new name.

New Forwarder Name:

# Removing Forwarders

If you want to remove a Forwarder, open its Settings page, and click the Remove Forwarder button.

**CommuniGate Pro**

| Introduction |
| Installation |
| SysAdmin |
| Network |
| **Objects** |
| Storage |
| E-mail |
| Real-Time |
| Access |
| Services |
| Directory |
| Clusters |
| Applications |
| Miscellaneous |
| Licensing |
| |
| WebMail |
| Pronto! |
| PBX |

**Objects  Domains  Accounts  Groups  Forwarders  Named Tasks  Chronos**

# Account Data

- **Creating a New Named Task**
- **Specifying Named Task Settings**
- **Renaming Named Tasks**
- **Removing Named Tasks**
- **Temporary Named Tasks**
- **Chatrooms**
- **Automatically Created Chatrooms**

The CommuniGate Pro Server provides special Domain Objects - "Named Tasks". When a Real-Time Signal is addressed to the Named Task name, the Server launches a Real-Time Application specified for that Named Task, and passes the Signal to that Application instance. When new Signals are directed to the same address, they are delivered to the same Application instance.

Named Tasks are used to implement collaboration services, such as XMPP-style "chatrooms".

# Creating a New Named Task

Named Tasks in any Domain can be created by the Server Administrator if the administrator Account has the Can Modify All Accounts And Domain Settings access right.

A Domain Administrator can create, remove, and rename Named Tasks only if the CanCreateNamedTasks access right is granted to the administrator Account.

To create a Named Task, create an Account or choose an existing one - the Named Task owner Account.
Use the WebAdmin Interface to open the Account Settings pages, then open the Call Settings page in the Real-Time section, and find the Named Tasks panel:

| Named Tasks | |
|---|---|
| **Name** | **Program Name** |
| projectx-talk | chatroom |
| | |

Enter the name of the Named Task to create and click the Create Named Task button.

The Server checks that there is no Account or other [Object](#) with the same name in this Domain, and creates a new Named Task.

---

# Specifying Named Task Settings

To specify Named Task Settings, click the Named Task name on the owner Account Settings page. The Named Task Settings page appears.

| | |
|---:|:---|
| Owner: | jksmith |
| Real Name: | |
| Program Name: | |

`RealName`
> A brief description of the Named Task.

`Program Name`
> The [Real-Time Application](#) name. The Named Task runs one instance of this Application.

To update the Named Task settings, click the Update button.

---

# Renaming Named Tasks

If you want to rename a Named Task, open its Settings page, fill the New Named Task Name field, and click the Rename Named Task button.

If there is no other Object with the same name as the specified new Named Task name, the Named Task is renamed and its Named Task Settings page should reappear on the screen under the new name.

| |
|:---:|
| New Named Task Name: |

---

# Removing Named Task

If you want to remove a Named Task, open its Settings page, and click the Remove Named Task button.

---

# Temporary Named Tasks

The CommuniGate Pro Server allows it users to create temporary Named Tasks. These Task are assigned a temporary name, and they are started immediately. When a temporary Named Task ends, its name is released and can be reused for other Objects in the same Domain.

Temporary Named Tasks can be used to create ad-hoc Instant Messaging chatrooms. When an XMPP or XIMSS client sends a request to retrieve a "unique room name", a Temporary Named Task running the `chatroom` Real-Time Application is created and started, and its name is reported to the client.

# Chatrooms

CommuniGate Pro implements XMPP-style chatrooms (multi-party Instant Messaging sessions) using the Real-Time Application `chatroom`.

The Application implements the chatroom according to the XEP-0045 specification.

The `chatroom` Application implements the following additional functions:

logging

the conversation log is stored in the owner Account file storage, as the `private/chatlogs/`*taskName*`/`*YYYY*`-`*MM*`.log` text files, where *taskName* is the chatroom (Named Task) name, *YYYY* is the year, and *MM* is the month number.

command line interface

a user can send a "groupchat" Instant Message starting with the `#cmd:` prefix. The rest of the message is interpreted as a command. The command is executed subject to the sending user current "role" and "affiliation". The command response is sent as an Instant Message with the `#cmdResult:` prefix to the command sender. The implemented commands are listed below.

chatroom merging

A CommuniGate Pro chatroom can be instructed to connect to some other chatroom (running on this or other CommuniGate Pro Server, or on some other XMPP server), as a participant.

When connected, the messages distributed inside this CommuniGate Pro chatroom are sent to the remote chatroom with a prefix specifying the sender nickname in this chatroom. In the same way messages distributed in the remote chatroom are distributed in the CommuniGate Pro chatroom with a prefix specifying the sender nickname in the remote chatroom.

A CommuniGate Pro chatroom can connect to several local and remote chatrooms at once, acting as a bridge between all of them.

The `chatroom` Application implements the following command line interface commands:

`help`

returns the supported command list.

`killroom`

disconnects all participants and terminated this Named Task application instance.

`kick` *nickName*
`kick [`*E-mail address*`]`

removes the specified participant from the chatroom.

This command is available to administrators and moderators only.

`voice` *nickName*
`voice [`*E-mail address*`]`

changes the specified participant "role" to `participant`, so the participant can post messages.

This command is available to administrators and moderators only.

`novoice` *nickName*
`novoice [`*E-mail address*`]`

changes the specified participant "role" to `visitor`, so the participant cannot post messages.

This command is available to administrators and moderators only.

`moderator` *nickName*
`moderator [`*E-mail address*`]`

changes the specified participant "role" to `moderator`.

This command is available to administrators only.

`member` *nickName*
`member [`*E-mail address*`]`

changes the specified participant "affiliation" to `member`.

This command is available to administrators only.

`admin` *nickName*
`admin [`*E-mail address*`]`

changes the specified participant "affiliation" to `administrator`.

This command is available to the Named Task owner only.

`guest` *`nickName`*
`guest [`*`E-mail address`*`]`

> changes the specified participant "affiliation" to `guest`.

> This command is available to administrators only.

`ban` *`nickName`*
`ban [`*`E-mail address`*`]`

> changes the specified participant "affiliation" to `outcast`, and removes the user from the chatroom.

> This user will not be able to connect to his chatroom again.

> This command is available to administrators only.

`msg` *`nickName message`*

> sends a private message *message* to the specified participant.

`announce` *`message`*

> sends a regular "groupchat" message *message* on the user behalf. This message will not be removed from the message history, but it can be replaced with the next `announce` command sent by any user. This command is available to administrators and moderators only.

`sysann` *`message`*

> sends a "groupchat" message *message* from the room itself. This message will not be removed from the message history, but it can be replaced with the next `sysann` command sent by any user. This command is available to administrators and moderators only.

`temp` *`message`*

> sends a regular "groupchat" message *message* on the user behalf. This message is not stored in the message history, and it is not recorded in the chatroom conversation log files. This command is available to administrators and moderators only.

`invite` *`userAddress`* `[` *`reason`* `]`

> sends an invitation to this chatroom to the *userAddress* user.

`join` *`roomAddress`*

> sends a join request to the *roomAddress* chatroom on the user behalf.

`merge` *`roomAddress`* `[` *`thisRoomNick`* `[` *`otherRoomNick`* `] ]`

> connects this chatroom as a participant to the *roomAddress* chatroom.

> *thisRoomNick* specifies the nickName of this chatroom inside the *roomAddress* chatroom. If *thisRoomNick* is not specified, this chatroom name (the Named Task name) is used.

> *otherRoomNick* specifies the nickName of the *roomAddress* chatroom in this chatroom. If *otherRoomNick* is not specified, this username part of the *roomAddress* is used. This command is available to administrators and moderators only.

---

# Automatically Created Chatrooms

Many XMPP-only "chat" implement handle "chatroom-only" domains, such as chats.mycompany.com. These servers automatically create a chatroom when an unknown name is addressed in their domain.

CommuniGate Pro Domains contain various Objects - Accounts, Aliases, Forwarders, Mailing Lists, Groups, and others, and creating chatrooms by user request is usually not desired: the Server or Domain administrator can create chatroom Named Tasks, while regular users can create "temporary chatrooms" with unique names, used for ad-hoc chatrooms.

To allow regular users to create chatroom Named Tasks by an attempt to connect to an non-existent name in the Domain, enabled the Auto-Create Chatrooms Domain Settings.
If a user of some Account in the same Domain tries to connect to a non-existent chatroom, a chatroom Named Task is created automatically, with this Account as its owner.
If an attempt to connect to a non-existent chatroom is done by a user of a different CommuniGate Pro Domain or by an external user, a chatroom Named Task is created only if the target Domain has the `chatmaster` Account. This Account is used as the newly created Named Task owner.

---

**Objects** **Domains** **Accounts** **Groups** **Forwarders** **Named Tasks** **Chronos**

# Chronos

- **Scheduled Tasks**
- **Configuring the Chronos Component**
- **Mailbox Cleaning and Archiving**

The Chronos component is used to schedule and perform automatic tasks for CommuniGate Pro Accounts.

These tasks include:

- Mailbox cleaning and archiving
- Calendar notifications
- Remote POP (RPOP) polling
- Remote SIP (RSIP) registrations

In the CommuniGate Pro Dynamic Cluster environment, the Chronos component scheduler runs on the Cluster Controller, distributing the Chronos tasks to all available backend Servers.

# Scheduled Tasks

Any CommuniGate Pro component can schedule a Chronos Task for a CommuniGate Pro Account. When a new Chronos Task is scheduled, the Account is placed into the Chronos process activation queue. At the scheduled time a Chronos processor thread opens the Accounts and performs the requested Task, re-scheduling the Account.

# Configuring the Chronos Component

You can use the WebAdmin Interface to configure the Chronos component. Open the General pages in the Settings realm, then open the Others page:

| Chronos |
| --- |
| Log Level:    Problems |

Log

Use this setting to specify the type of information the Chronos component should put in the Server Log. Usually you should use the `Failure` (unrecoverable problems only), `Major` (Chronos task completion reports), or `Problems` (failures, completion reports, and non-fatal errors) levels. When you experience problems with the

Chronos component, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case the Chronos processing internals will be recorded in the System Log. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.
The Chronos component records in the System Log are marked with the `CHRONOS` tag.

# Mailbox Cleaning and Archiving

The Chronos component can automatically clean and archive the Account Mailboxes.

If the Archive Messages after [Account setting](#) is set to any value other than Never, then all messages from the `INBOX` and *SentItem* Mailboxes which are older than the specified time period are moved to the
`Archive/YYYY-MM`
Mailboxes, where *YYYY* is the year number and *MM* is the month number of the moved messages "internal date".
The *SentItem* Mailbox name is specified with the Account Preferences.

If the Delete Messages after [Account setting](#) is set to any value other than Never, all messages from all Account Mailboxes which are older than the specified time period are periodically deleted. **This process does not apply to Mailboxes with a non-empty [Mailbox Class](#) - i.e. to Calendar, Contacts, Tasks, Notes, and other non-mail Mailboxes.**

**The Archive Mailboxes created to copy `INBOX` and *SentItem* messages are created with a non-empty Mailbox Class.**

| **Storage** | **Mailboxes** | **Files** | **External** |

# Objects

- **Mailboxes**
- **Groupware Items**
- **File Storage**

Each CommuniGate Pro Account stores various data: E-mail messages, Calendar appointments, Contacts, files, SIP client registrations, subscriptions, etc. This section explains how this information is organized and how it is stored in the CommuniGate Pro Accounts.

# Mailboxes

A Mailbox is the basic *storage unit*: messages sent to Accounts are stored in Account Mailboxes. Messages can be read from Mailboxes, they can be marked with various flags, they can be copied to other Mailboxes, and they can be removed from Mailboxes.

Each Account can have one or several Mailboxes. The INBOX Mailbox is special: it exists in every Account, and it is used to store incoming messages. The INBOX Mailbox is created automatically when an Account is created. A user cannot remove the INBOX Mailbox, but a user can rename it. In this case, a new empty INBOX is immediately created.

CommuniGate Pro allows administrators to create *single-mailbox* Accounts. These Accounts contain only the INBOX Mailbox.

The CommuniGate Pro Server provides access to Account Mailboxes via POP, IMAP, XIMSS WebUser Interface, and other modules.

CommuniGate Pro Mailboxes can have various formats. Administrators and users can select the Mailbox format when they create a new Mailbox.

See the [Mailboxes](#) section for more information about CommuniGate Pro Mailboxes.

The CommuniGate Pro Server can be integrated with other systems, so the CommuniGate Pro Accounts can utilize the mailbox storage of those systems. See the [External Storage](#) section for more information.

# Groupware Items

Account Groupware items, such as Calendar and Task items, Contacts, Notes are stored as E-mail messages. Usually they are stored in the Mailboxes dedicated for a specific Groupware function (such as Calendar, Task, Contact, Notes "folders") and assigned proper Mailbox Classes. But there is no limitation on the Mailbox Classes, and any E-mail or Groupware item can be stored in any Mailbox, of any Mailbox Class.

# File Storage

Each Account has its own File Storage tha can store any set of files and file directories. See the [File Storage](#) section for the details.

CommuniGate **Pro**

| Storage | **Mailboxes** | Files | External |

# Mailboxes

- **Mailbox Names**
- **Message Flags**
- **Mailbox Access Rights**
- **Mailbox Formats**
  - **The Text Mailbox (.mbox) Format**
  - **The MailDir Mailbox (.mdir) Format**
  - **The Sliced Mailbox (.mslc) Format**
- **Mailbox Classes**
- **Special Mailboxes**
- **Locked Mailboxes**
- **Creating Mailboxes**
- **Mailbox Subscription**
- **Mailbox Aliases**
- **Simultaneous Access**
- **Foreign and Public Mailboxes**
- **Legacy Mailboxes**

CommuniGate Pro Accounts contain one or several Mailboxes. Each Mailbox has its own unique name and it can contain zero or more messages. The POP, IMAP, MAPI, XIMSS, WebUser Interface, and Real-Time Application modules provide access to Account Mailboxes.

Several storage formats can be used for CommuniGate Pro Mailboxes. A multi-Mailbox Account can contain Mailboxes stored in different formats.

Each Account always has the INBOX Mailbox. Any message delivered to a CommuniGate Pro Account is stored in its INBOX Mailbox - unless some Automated Processing Rules instruct the Server to store the message in a different Mailbox.

# Mailbox Names

When an Account is created, its INBOX Mailbox is automatically created. The System and/or Domain Administrator can specify additional Mailboxes to be created at that time.

A user can create a Mailbox using an IMAP, MAPI, or XIMSS mailer application or using the WebUser Interface.

Mailboxes can be "nested": for any Mailbox "A" you can create a sub-Mailbox "B" - in the same way as you can created a file directory inside some other file directory. The CommuniGate Pro Server uses the slash (/) symbol as the hierarchy separator: INBOX/important is the name of the sub-Mailbox important "inside" the INBOX Mailbox.

CommuniGate Pro allows you to store messages in some Mailbox x and at the same time you can create sub-Mailboxes x/Y, x/z for that Mailbox. This feature is implemented by providing two "invisible" Mailbox entities - one for storing messages, one - for serving as a "directory" for the nested Mailboxes. The "directory" entity is created automatically, as soon as you try to create the first sub-Mailbox. You can, though, create the "directory" entity without

creating the "mail storage" entity: use the `ABCDEF/` name as the new Mailbox name to create only the directory entity with the `ABCDEF` name. The name `ABCDEF` will be listed, but will not be "selectable" - and you will not be able to store messages in the `ABCDEF` Mailbox. You can later create the regular `ABCDEF` Mailbox and the "storage" entity for your `ABCDEF` Mailbox name will be added.

It is impossible to delete the INBOX Mailbox. You can rename the INBOX Mailbox, though. In this case a new empty INBOX Mailbox will be created automatically.

Mailbox names are case-sensitive. Some file systems (NTFS, for example) provide case-insensitive file naming conventions. When these file systems are used for CommuniGate Pro Account/Mailbox storage, the Mailbox names are still case-sensitive, but you cannot create two Mailboxes with names that differ in case only. The INBOX Mailbox name is an exception: it is always a case-insensitive name.

---

# Message Flags

Messages in Mailboxes have individual flags. These flags can be set when the message is being stored in the Mailbox, and they can be updated using Mailbox access protocols and methods, such as [IMAP](#), [MAPI](#), [XIMSS](#), [WebUser Interface](#), [Real-Time Applications](#).

Some flags are set automatically, even when the access protocol used does not support flag modification. For example, the Seen flag is set automatically when the message is being read using the [POP](#) protocol RETR command.

Several components (such as [Automated Rule](#), [CG/PL](#) programs, etc.) can access message flags by name. They can also use "negative names" to instruct the server to reset a certain flag or to look for messages that do not have that flag set.

The following table lists the supported message flags along with their IMAP and Negative names:

| Name | Description | IMAP Name | Negative Name |
|---|---|---|---|
| Seen | This flag is set when the message was read by a client. It can be set automatically as a result of certain Mailbox access operations, and it can be set and reset explicitly with mail client applications. | \Seen | Unseen |
| Read | *same as Seen* | | Unread |
| Answered | This flag is set when a reply was sent for this message. This flag is explicitly set and reset with mail client applications. | \Answered | Unanswered |
| Flagged | This flag is set to attach a "flag" to the message (for example, a mail client can show this message to the user as an important one). This flag is explicitly set and reset with mail client applications. | \Flagged | Unflagged |
| Draft | This flag is set for messages that have not been sent yet. It tells a mail client that it can open and edit this message. This flag is explicitly set and reset with mail client applications. | \Draft | Undraft |
| Deleted | This flag is set for messages that were marked for deletion. Some mail clients allow users to mark some Mailbox messages first, and then delete ("expunge") all marked messages from the Mailbox. This flag is explicitly set and reset with mail client applications. | \Deleted | Undeleted |
| Redirected | This flag is set when a copy of the message was sent (redirected) to someone. This flag is explicitly set and reset with mail client applications. | $Forwarded | NotRedirected |
| MDNSent | This flag is set when an MDN ("read report") for the message has been sent. This flag helps mail clients to send only one MDN report for each message. This flag is explicitly set and reset with mail client | $MDNSent | NoMDNSent |

| | | | |
|---|---|---|---|
| | applications. | | |
| Hidden | Messages with this flag set are visible only to the Mailbox Account owner and to those users who have the Admin Access Right for this Mailbox.<br>This flag allows users to grant access to their Mailboxes to others while keeping certain messages private (hidden). | $Hidden | NotHidden |
| Service | Messages with this flag set are not visible to IMAP or POP clients. MAPI clients can use this flag to create service items invisible to users (such as Mailbox forms). | $Service | NotService |
| Media | If this flag is set, the message is treated as containing some "media" (audio/video) data. | $Media | NotMedia |
| Junk | If this flag is set, the message is treated as "junk" (spam). | Junk | NotJunk |
| Label1 | Custom flag. | $Label1 | NotLabel1 |
| Label2 | Custom flag. | $Label2 | NotLabel2 |
| Label3 | Custom flag. | $Label3 | NotLabel3 |

# Mailbox Access Rights

The CommuniGate Pro Server maintains an Access Control List (ACL) for every Mailbox it creates.

The Access Control Lists are used to control the Foreign Mailbox Access feature that allows Account users to access Mailboxes in other Accounts.

A Server Administrator with the All Domains access right has all access rights to all Mailboxes in all Server or Cluster Accounts.

Domain Administrators with the CanViewMailboxes access right have all access rights for all Mailboxes in their Domains.

The Mailbox Account owner can grant certain limited access rights to other users, using the Access Control Lists.

The following Mailbox access rights are supported:

l (Lookup)
>   If you grant a user the Lookup access right, that user will be able to see this Mailbox when it asks the Server to list all Mailboxes in your Account.

r (Read/Select)
>   If you grant a user the Read access right, that user will be able to open (select) this Mailbox and see (read) the messages in this Mailbox.

s (Seen)
>   If you grant a user the Seen access right, that user will be able to mark messages as read (seen). Usually a message is automatically marked as seen when a user reads it. But if this access right is not granted to a user reading the Mailbox, the Mailbox message "seen" status will not be changed.

w (Write/Flags)
>   If you grant a user the Write access right, that user will be able to set message flags: i.e. to mark messages as answered or "flagged", and to reset the message flags.

d (Delete)
>   If you grant a user the Delete access right, that user will be able to mark messages as deleted and to compress

the Mailbox, removing all its messages marked as deleted.

i (Insert)
      If you grant a user the Insert access right, that user will be able to append messages to this Mailbox and to copy messages from other Mailboxes into this one.

p (Post)
      This access right is not used by modern mailers.

c (Create)
      If you grant a user the Create access right, that user will be able to create new Mailboxes "inside" this Mailbox.

a (Administer)
      If you grant a user the Administer access right, that user will be able:

- to modify the Mailbox ACL
- to modify the Mailbox meta-data (such as the Mailbox Class)
- to see Hidden Mailbox messages

When a sub-Mailbox is created, it inherits the ACL of the outer (parent) Mailbox. This means that if you create the `INBOX/sales` Mailbox, it is created with the same ACL as specified for the `INBOX` Mailbox.

To delete a foreign Mailbox, a user should have:

- the Create access right for the outer (parent) Mailbox, and
- the Delete access right for the specified Mailbox

To rename a foreign Mailbox, a user should have:

- the Create access right for the outer (parent) Mailbox of the original Mailbox, and
- the Delete access right for the specified original Mailbox, and
- the Create access right for the outer (parent) Mailbox of the new Mailbox (Mailbox name)

To create a foreign Mailbox, a user should have the Create access right for the outer (parent) Mailbox of the Mailbox to be created.

When the target Mailbox is a "top" one in an Account, and thus there is no outer (parent) Mailbox, the "CreateMailboxes" Account Access Right is checked instead.

The Mailbox Access Control Lists can be set and modified using the WebUser Interface, a XIMSS, a MAPI, or a decent IMAP client.

---

# Mailbox Formats

CommuniGate Pro stores received messages in Account Mailboxes. The server supports several Mailbox formats, and the Mailbox type is defined by the Mailbox file (or directory) name extension.

For single-Mailbox Accounts, the Mailbox type is specified when the Account is created.

Each multi-Mailbox Account has a setting that specifies the default type for all new Mailboxes created in this Account. A user can explicitly specify the Mailbox type creating a Mailbox in a multi-Mailbox Account: if the Mailbox name is specified as `name.extension`, then the Mailbox *name* of the *extension* type is created.

# The Text Mailbox (`.mbox`) Format

The Mailbox files with this extension store messages in the legacy BSD mailbox format. Each message in the Mailbox is preceded with the a *From-line*:

`From <>(flags-UID-origUID) time stamp`

This is the same format as one used in legacy mail systems, but with a "comment" added after the return-path part. The .mbox format remains compatible with legacy applications (local mailers), and at the same time it allows the CommuniGate Pro Server to store the required message information (message status flags, the unique Mailbox message ID, and, optionally, the "permanent" message UID).

If a Mailbox file has been copied from an old system, or when it is used as an [Legacy INBOX](#) and old applications can add messages to this Mailbox, some messages may have no "comment;" part. CommuniGate Pro allows a user to work with such messages, but it does not store message flags if they were modified, and it does not remember the message UIDs between sessions. The simplest solution is to copy such messages to a different Mailbox and then copy them back to the original Mailbox - the copy operation places the correct information into the *From-line*.

When a message is being stored in the `.mbox`-type Mailbox, all message lines are checked. If there is an empty line followed with the line starting with the letters `From`, the '>' symbol is inserted before the letter `F`.

The Text Mailboxes become less effective as their size grows. When a Text Mailbox is being opened, it has to be parsed, in order to detect message boundaries and retrieve the UID, flags, and other per-message information. When some messages are being deleted from the middle of a Text Mailbox, the Server has to copy the remaining messages data, compressing the Mailbox. To make these processes more efficient, the CommuniGate Pro server can deal with Mailbox data in large chunks. A special semaphore object limits the number of buffers allocated for large Mailbox processing. Changing this parameter can change the overall large Mailbox access (you may want to increase or decrease it, depending on the OS and file system you use).

To improve Text Mailbox opening speed, the CommuniGate Pro can maintain a Mailbox index (`.bdx`) file alongside the Text Mailbox Mailbox file. If the index file exists, the Server reads it instead of parsing the entire Mailbox file. CommuniGate Pro automatically creates an index file when it the Mailbox file size exceeds the specified limit. The Server removes the index file if the Mailbox becomes smaller than that limit.

The Index file is created when any message in the Mailbox is modified or deleted. If new messages have been added to the Mailbox, but the Mailbox has not been opened, or it has been only read without any flag modification, the Index file may not be created.

To store Message Attributes, special "invisible" messages are created in the Text Mailbox.

Use the WebAdmin Interface to specify the Text Mailbox Manager settings. Open the General pages in the Settings realm, and find the Text Mailbox Manager panel on the Others page:

| Text Mailbox Manager | |
| --- | --- |
| Concurrently used large buffers: | 20 |
| Index Mailboxes larger than: | 300K |

`Concurrently used large buffers`
> Use this setting to specify how many concurrent operations (parsing, deletion) the Text Mailbox Manager can perform on large Mailboxes.

`Index Mailboxes larger than`
> Use this setting to specify the minimal size for Mailboxes that need indexing.

# The MailDir Mailbox (`.mdir`) Format

Mailboxes with this extension are file directories. Each Mailbox message is stored as a separate file in the Mailbox directory.

The message file name has the following format:
`iiiiOoooo-flags-timestamp-nLines`
where *iiii* is the message unique ID, O*oooo* is the message permanent UID (optional), *flags* are the message status flags, the *timestamp* is the message *internal time stamp* - the time (GMT) when the message was added to the Mailbox, in the `yyyymmddhhmmss` format, and *nLines* is the number of text lines in the message.

**Note:** On the Unix platforms, the MailDir Mailboxes implement the *shared storage model*: if the same message is directed to many Accounts/Mailboxes, only one message file is created, and a *hard link* to that file is placed into each Mailbox directory.
When a message is removed from all Mailboxes, the file is automatically deleted by the OS.
This feature is disabled for the Accounts that have the Erase Deleted Messages Setting enabled.

To store Message Attributes, special files with `Ooooo` names are created inside the Mailbox directory.

**Note:** most freeware mail systems use either the mbox-like or mdir-like formats, and designers of those systems make various claims about the advantages of the formats they have selected. It is important to remember that:

- CommuniGate Pro does not use OS or file system features (locks) to provide multi-access to Mailboxes. CommuniGate Pro Mailboxes in all formats can be accessed by several clients and components at the same time, and all synchronization is implemented inside the CommuniGate Pro Mailbox Manager that works in the same way for all Mailbox formats.
- CommuniGate Pro uses efficient mechanisms to parse Mailboxes, so many claims about various Mailbox formats being 'slower' or 'faster' than other formats usually do not apply to CommuniGate Pro installations.
- CommuniGate Pro allows users to create Mailboxes in different formats, even within the same Account.

**Note:** the Text format is more efficient than the MailDir format in most cases, this is why this format is used as the default one. The MailDir format is recommended only for those Mailboxes that contain many (100 or more) large (1MB or more) messages. If a user has a `Proposals` Mailbox where she stores all messages with attached documents, each 0.5-5MB in size, then this Mailbox may work faster if it is created in the .mdir format.

## The Sliced Mailbox (`.mslc`) Format

Mailboxes with this extension are file directories. These directories contain `dataNNNN` files, each file contains one or more messages, in the format similar to one used in Text Mailboxes.

The `index.bdx` file contains the mailbox index, and includes information for all messages stored in all data files.

As new messages are added to such a Mailbox, they are either appended to the existing data files, or stored in a newly created data file, depending on the message size and the size and number of messages stored in the existing data files.

When messages are removed, data files can be merged to keep the total number of data files low.

---

# Mailbox Classes

Each Mailbox can have a Class attribute. This attribute specifies the type of the information this Mailbox is created for: Calendar, Contacts, Tasks, Notes, etc. If a Mailbox does not contain the Class attribute, it means that it is created to store regular E-mail messages.

The Mailbox Class does not restrict the types of data that can be stored in the Mailbox: E-mail and Contacts messages can be stored in Mailboxes with the Tasks Class, Notes messages can be stored in Calendar Class Mailboxes, etc. The

Mailbox Class information is used with the advanced user interfaces (WebUser, MAPI) to present the Mailbox content in the proper format.

When a Mailbox is created with an advanced client interface, the interface can set the Mailbox Class. Mailbox Classes can also be updated using the CommuniGate Pro [CLI/API](#).

---

# Special Mailboxes

Some Mailboxes have special meanings.

The `INBOX` Mailbox receives all incoming E-mail messages, unless these messages are explicitly directed to other Mailboxes with [Rules](#) or using [Direct Mailbox Addressing](#).
When an Account is created, its `INBOX` Mailbox is created automatically.

Other special Mailboxes do not have fixed names. The Account user or an administrator can specify or modify the name used for any special Mailbox. These names are stored in Account Preferences.

The CommuniGate Pro Server recognizes Special Mailbox names (starting and ending with the `$` symbols) as references to special Mailboxes. This feature allows client applications to access special Mailboxes in any Account, without being reconfigured to use the actual name of those special Mailboxes. These Special names do not show up in the Account Mailbox lists.

Trash
> This Mailbox is used to store messages to be deleted. Clients can move messages from other Mailboxes to this Mailbox, using it as a "Tash can". Special name: `$Trash$`

Sent Messages
> This Mailbox is used to store copies of sent E-mail messages. Special name: `$Sent$`

Drafts
> This Mailbox is used to store E-mail message drafts. Special name: `$Drafts$`

Junk
> This Mailbox is used to store junk E-mail messages. Special name: `$Junk$`

History
> This Mailbox is used to store "log"-type messages, such as copies of the IM sessions. Special name: `$History$`

Default Calendar
> This Mailbox is used as the Account main Calendar Mailbox. This Mailbox is expected to be of the [Calendar class](#). Special name: `$Calendar$`

Default Task List
> This Mailbox is used as the Account main Tasks Mailbox. This Mailbox is expected to be of the [Tasks class](#). Special name: `$Tasks$`

Default Contacts
> This Mailbox is used as the Account main Contacts Mailbox. This Mailbox is expected to be of the [Contacts class](#). Special name: `$Contacts$`

Default Notes
> This Mailbox is used as the Account main Notes Mailbox. This Mailbox is expected to be of the [Notes class](#). Special name: `$Notes$`

# Locked Mailboxes

Each Mailbox can have the Locked attribute. If this attribute is set, the Mailbox cannot be deleted or renamed.

A locked Mailbox can be deleted or renamed together with its parent Mailbox, if the parent Mailbox itself is not locked.

You can specify the Locked attribute for the Mailboxes created using the Account Template. The Mailbox Locked attribute can also be updated using the CommuniGate Pro CLI/API.

# Creating Mailboxes

Every Account has a setting that specifies the default format for new Mailboxes that can be created in this Account.

The Account user can explicitly specify the storage format for a new Mailbox by adding the format extension to the new Mailbox name. If a user tells the CommuniGate Pro Server to create the `newmailbox.mdir` Mailbox, the `.mdir`-formatted Mailbox `newmailbox` is created.

# Mailbox Subscription

The CommuniGate Pro Server allows an Account user to *subscribe* to some Mailboxes. The Account Mailbox subscription is a simple list of Mailbox names. This list is not used by the Server itself - the Server just stores one subscription list for each Account.

Many IMAP mailers use the Account subscription list and show only the Mailboxes the Account is subscribed to. The WebUser Interface can also be configured to show only the subscribed Mailboxes.

You can modify the Account subscription either via a decent IMAP mailer, or using the WebUser Interface.

You can use the Account Mailbox subscription to make some not-so-decent IMAP mailers access foreign Mailboxes: make sure that your IMAP client is configured to use the Account Mailbox subscription, and add the desired foreign Mailbox name into the subscription list.

**Note:** Some IMAP mailers tend to rebuild Account subscription lists: they empty the subscription, and then subscribe you to all Mailboxes in your own Account.

The Account Mailbox subscription is stored in the Account `.info` service file.

# Mailbox Aliases

Many IMAP clients (such as Microsoft Outlook and Outlook Express) and AirSync clients (such as Windows Mobile, Apple iPhone, Nokia PDAs, etc.) cannot handle foreign Mailboxes directly, and they cannot use the Account Mailbox subscription to access foreign Mailboxes.

Mailbox Aliases can be used to let these clients access foreign Mailboxes.

A Mailbox Alias is a name associated with some [foreign] Mailbox name. For example, you can create a Mailbox

Alias `salesBox` for the `~sales/INBOX` Mailbox name. You will see the `salesBox` Mailbox in your client application, but in reality this will be the `INBOX` Mailbox in the `sales` Account.

Mailbox Aliases can be created only on the topmost level of the Account Mailbox hierarchy, that means that the Mailbox Alias name cannot contain the slash (`/`) symbol.

A Mailbox Alias provides access to the Mailbox it is associated with, and to all its sub-mailboxes (subject to [Access Control List](#) restrictions).

A Mailbox Alias can contain just a foreign Account name (`~accountName`). Such an Alias provides access to all accessible Mailboxes in that foreign Account. The Mailbox Alias itself is presented as an unselectable Mailbox name.

Sample configuration:

> The owner of the Account `chief` has granted "lookup" and other access rights for his Mailboxes `INBOX` and `Pending` to the `assistant` Account.
>
> The user `assistant` has created the Mailbox Alias `boss` pointing to `~j.smith`.
>
> When the user `assistant` connects to her Account using any IMAP or XIMSS client, or the WebUser Interface, she sees all her own Mailboxes, the unselectable Mailbox `boss`, and also the `boss/INBOX` and `boss/Pending` Mailboxes.
>
> If the user `j.smith` creates a new Mailbox `Urgent` in his Account and grants access rights for that Mailbox to the `assistant` Account, the user `assistant` will immediately see the new Mailbox as the `boss/Urgent` Mailbox.

# Simultaneous Access

The CommuniGate Pro Server allows several client applications to connect, open the same Mailbox, and read and modify the Mailbox data at the same time.

The CommuniGate Pro *multithreaded* design allows the Server to synchronize client activities without using OS-level *file locks* and it does not require a client to wait till all other clients close the Mailbox.

Simultaneous Access means that:

- several clients can have simultaneous access to the same Mailbox
- new messages can be added to a Mailbox while mailer clients are working with that Mailbox
- messages can be deleted from a Mailbox while mailer clients are working with that Mailbox

Clients accessing the same Mailbox can use the same or different Mailbox access protocols - [POP](#), [IMAP](#), [MAPI](#), [AirSync](#), [WebUser](#), or [XIMSS](#) Interface.

Simultaneous Access is supported for all [Mailbox types](#) implemented in the CommuniGate Pro software.

This feature allows you to work with your Mailbox from several workstations and devices, and it lets a group of people (i.e. the sales department) process messages in one centralized Mailbox.

# Foreign and Public Mailboxes

The CommuniGate Pro allows an Account user to access Mailboxes in other Accounts.

Access to these foreign Mailboxes (also called shared Mailboxes) is controlled via the [Mailbox Access Rights](#).

To access a Mailbox in a different Account, the Mailbox name should be specified as `~accountname/mailboxname`. For example, to access the `INBOX` Mailbox in the Boss Account, the Mailbox name should be specified as `~Boss/INBOX`.

If there are several local Domains on the Server, Mailboxes in a different Domain can be accessed by specifying full Account names. To access the `LIST/reports` Mailbox in the Account ListMaster in the client.com Domain, the Mailbox name should be specified as `~ListMaster@client.com/LIST/reports`.

Account names specified after the "~" sign are processed with the [Router](#), so Account Alias names can be used instead of the real Account names, and all Routing Table rules are applied.

Very often Foreign Mailboxes are used:

- to let a secretary view and mark messages in your INBOX;
- to let several sales persons see and process a single "sales maildrop" - the INBOX of the `sales` Account;
- to let several engineers see and process a single "technical support maildrop" - the INBOX of the `support` Account.

CommuniGate Pro can provide "public" Mailboxes, too. This can be done by creating an Account `public`, and assigning public Access rights to its Mailboxes. Usually, each group of public Mailboxes is managed by some administrator, who is not required to be a CommuniGate Pro administrator.

A CommuniGate Pro Server administrator should create the `public` Account, log into that Account using the WebUser Interface or a decent IMAP client, create some public Mailboxes, and grant administration rights to regular users that will administer these public Mailboxes. Those users will then grant access rights to other users, create sub-Mailboxes, and perform other administrative tasks.

For example, a public Mailbox administrator can use [Automated Rules](#) to copy certain incoming messages directly into some public Mailbox.

Some IMAP clients (such as Microsoft Outlook and Outlook Express), and most AirSync clients do not support foreign Mailboxes at all. To let those clients access shared Mailboxes in other Accounts, [Mailbox Aliases](#) can be used.

---

# Legacy Mailboxes

On some systems users have direct (login) access to the mail server computer, and some of them get used to *Local Mailers* - `mail`, `elm`, and others. *Local Mailers* do not use any network protocol to access mailboxes. Instead, those programs read and modify mailbox files directly, via the file system.

The CommuniGate Pro allows you to create Accounts with *Legacy* INBOX Mailboxes. These Mailboxes are stored not inside the CommuniGate Pro *base directory*, but in the system directory known to the legacy mailer applications.

Since these INBOX files can be read and modified directly, bypassing the CommuniGate Pro protocols and modules, the Server needs to synchronize its activity with legacy mail applications using OS *file locking* features - either FileLevel locks or FileRange locks.

On Unix systems the FileLevel locks are known as `flock` operations, and RangeLevel locks are known as `fcntl` operations. Check with your OS manual to see which method the legacy mailers use on your system, and configure the CommuniGate Pro Server to use that method. For systems that support only one file locking mechanism (MS Windows, Sun Solaris, and some other systems), selecting either method selects that mechanism.

You should use Legacy Mailboxes only when absolutely necessary, because:

- access to Legacy Mailboxes is less effective as it consumes resources needed for OS *file locking*.
- *local mailer* applications do NOT synchronize correctly, and **there is always a chance that a local mailer destroys a mailbox**. If some of your users have to work via local mailers, warn them about this fact, and ask them to avoid using *local mailers* and modern (protocol-based) mailers at the same time. These bugs in most local mailers do not show up if messages are only added to the mailbox when a local mailer is active. Local mailers may corrupt mailboxes if messages are deleted from a mailbox during the time a local mailer is active.

If you have to support Local Mailer compatibility for all or some Accounts in a Domain (usually - in the Main Domain), you should specify the [Legacy INBOX](#) settings for that Domain.

When you create an Account that has an Legacy INBOX, the Server checks if the Account INBOX file already exists in the specified location and creates one if the Mailbox file is absent.

When you delete an Account that has an Legacy INBOX, the Server does NOT remove the INBOX Mailbox file.

---

| Storage | Mailboxes | **Files** | External |
|---------|-----------|-----------|----------|

# File Storage

- **Special Files and Folders**
- **Virtual Files and Folders**
- **File Attributes**
- **File Access Rights**
- **Shared Private Files**
- **HTML-based Management**
- **HTTP-based Management**
- **HTTP Access to File Storage**
- **FTP Access and Management**
- **WebDAV Access and Management**
- **Foreign File Access**
- **File Subscription**

CommuniGate Pro Accounts contain a *File Storage* - a set of HTML, JPEG, and other files. File Storage can contain directories which can contain files and other (*nested*) directories.

The CommuniGate Pro HTTP module can be used to access and to manage the Account File Storage. This feature makes an Account File Storage act as a *personal Web site*.

The CommuniGate Pro FTP module, the TFTP module, and the HTTP module WebDAV extension can be used to access and to manage the Account File Storage.

The CLI API can be used to access and to manage the Account File Storage.

The total number of files and folders and the total size of all File Storage files is limited with the Account Settings.

The Account File Storage is used:

- to store message attachments.
- to store audio prompts and other files used in Real-Time Applications.
- to store various Account-level Logs (such as call logs).

# Special Files and Folders

Certain File Storage names have special meanings.

*empty*

An empty-name file is retrieved via HTTP when no file name is specified in the Access URL, such as

`http://server:port/~username/`, or `http://server:port/~username`, or `http://server:port/~username/subfolder/`

In this case, a file with a predefined name is retrieved. This predefined name is set as an Account Setting, and is usually set to `default.html`. So, an HTTP retrieval request for the
`http://server:port/~username/subfolder/` URL produces the same result as the

```
http://server:port/~username/subfolder/default.html
```
request.

freebusy.vfb

This text file contains the user Free/Busy information. When the Account Main Calendar is modified with the CommuniGate Pro [MAPI](#) module, [XIMSS](#) interface, or [AirSync](#), [CalDAV](#), or the [WebUser Interface](#) module, the file is deleted.

When this file is being accessed and it does not exist, the Server opens the Account Main Calendar, builds the Free/Busy information, and stores the information in the `freebusy.vfb` file.

If the Free/Busy information cannot be built (for example, if no Main Calendar Mailbox exists in the Account), the [HTTP](#) module generates an empty Free/Busy dataset and sends it to the client.

*any_name*.meta

these names are reserved and cannot be used.

private/logs/calls-*yyyy-mmm*

These text files are created and filled with the [Signal Dialog](#) objects. *yyyy* is a 4-digit year number, and *mmm* is a 3-letter month name.
Each file record has tab-delimited fields and contains information about one call made from or to the Account.

private/IM/*userName@domainName*.log

These text files contain [Instant Messages](#) send to and received from the *userName@domainName* address.
Each Instant Message is stored as one text line, with the following tab-delimited fields:

- time stamp (GMT): *YYYYMMDDThhmmsss*
- direction: `<` for incoming IMs, `>` for outgoing IMs
- deviceID: *[string](#)* - for "groupchat" messages - the sender or receiver device ID, an empty field otherwise
- body: *[string](#)* - the message body text, as a String object [presentation](#).

**Note:** names starting with the tilda (~) symbol are used to specify [Foreign Files](#), and, as a result, this symbol cannot be used as the first symbol of a file name.
Many Microsoft® products use names starting with the tilda symbol for temporary or service files. To avoid the problem, always use those products with Account Storage subdirectories, and not with the topmost Storage directory.

# Virtual Files and Folders

Virtual names do not specify actual files or folders in the File Storage, but they can be used to retrieve certain information.

index.wssp

This name is used for [HTML-based File Storage management](#). Access to this resource requires authentication.

freebusy.wssp

This name is used to retrieve formatted Free/Busy information. The actual data is retrieved from the `freebusy.vfb` file (see above).

pubcal/*calendarName*.ics

These names are used to retrieve the contents of the specified Account Calendar Mailbox in the iCalendar format. This virtual files can be used to "publish" the Calendar information. The user must have the right to Select the specified Mailbox.

$DomainSkins, $ServerSkins, $ClusterSkins

These virtual directories provide access to the Domain-wide or to the Server-wide/Cluster-wide Skins. Each Skin is presented as a subdirectory, with the name `$unnamed$` used for the "unnamed" Skins.
The Account must have the proper Access Rights to see and manage these directories.

`$DomainPBXApp`, `$ServerPBXApp`, `$ClusterPBXApp`
These virtual directories provide access to the Domain-wide or to the Server-wide/Cluster-wide Real-Time Application Environments. Language variants are presented as subdirectories.
The Account must have the proper Access Rights to see and manage these directories.

# File Attributes

Each file and file directory can have an set of attributes or meta-information.

For example, the `Betty.jpeg` file contains meta-information such as the location where the photo was taken, comments, etc.

Each attribute is an XML element.

Some attributes are "protected" - they can be modified only by the Account owner, the System or Domain Admin or if the user is granted the "Administer" Access Right to that file or file directory.

# File Access Rights

The CommuniGate Pro Server maintains an Access Control List (ACL) for every Storage file or file directory. This list is stored as an `<ACL>` protected File Attribute.

The Access Control Lists are used to control the Foreign File Access feature that allows Account users to access File Storage in other Accounts.

All files and file directories in an Account File Storage located outside the `private` directory are open for "list" (directories) and "read" operations for any Account user, as well as for non-authenticated users. For example, these files can be accessed via unauthenticated HTTP requests, and they can be used as a Personal Web Site.

The Account owner has all access rights to all Account Storage files and directories.

A Server Administrator with the All Domains access right has all access rights to all files in all Server or Cluster Accounts.

Domain Administrators with the `CanViewWebSites` access right have all access rights for all files in their Domain Accounts.

The Account owner can grant certain limited file access rights to other users, using the Access Control Lists.

The following File Access Rights are supported:

`l` (List, file directories only)
If you grant a user the List access right, that user will be able to see the content of this file directory.

`d` (Delete, file directories only)
If you grant a user the Delete access right, that user will be able to remove files and empty file sub-directories from this file directory.

**r** (Read)

If you grant a user the Read access right to a file, that user will be able to read this file and its unprotected attributes. If you grant a user the Read access right to a file directory, that user will be able to read all files in this directory for which the Access Control Lists are not specified.

**w** (Write)

If you grant a user the Write access right to a file, that user will be able to write to this file and to modify its unprotected attributes. If you grant a user the Write access right to a file directory, that user will be able to write to all files in this directory for which the Access Control Lists are not specified, and the user will be able to create new files in this directory.

**a** (Administer)

If you grant a user the Administer access right, that user will be able to modify the file ACL and other "protected" File Attributes.

When a file directory is created, the ACL of the outer directory (if any) is copied to the newly created directory.

---

# Shared Private Files

Read access to files and List access to directories inside the `private` directory can be granted to other CommuniGate Pro users and external "guests", using the protected `<accessPwd>` File Attribute.

Each `<accessPwd>` attribute should have a `<key/>` element containing a random string - the *access-password*. It is recommended to add `<EMail/>` element(s) with the E-mail address(es) of the users to whom this *access-password* has been sent.

Example:

```
<accessPwd>
  <key>dyf984897498ih12ui3u-3y887</key>
  <EMail realName="User Name">user1@domain1.dom</EMail>
  <EMail>user2@domain2.dom</EMail>
</accessPwd>
```

When this attribute is added to the `private` directory, it enables an alternative access path to all files within that directory and its subdirectories:

    protected/pwd/*access-password*/

If this attribute is added to the `private/dir1/dir2` directory, it enables an alternative access path to all files within that directory and its subdirectories:

    protected/dir1/pwd/*access-password*/dir2/

If the XML attribute listed in the example above is added to the `private/dir1/dir2` directory file attributes, the user can compose a Personal Site URL using the alternative path:

    http://server:port/~username/protected/dir1/pwd/dyf984897498ih12ui3u-3y887/dir2/file.name

The user then can send this URL to `user1@domain1.dom` and `user2@domain2.dom` and other addresses. The recipients can access this `file.name` file by following this link.

The user can revoke this access right by removing this `<accessPwd/>` attribute.

Alternative file paths can be used in FTP and TFTP protocols, and in all other CommuniGate Pro components that access the Account File Storage.

# HTML-based Management

Users can manage their Account File Storage using a Web browser. There are two methods to access the File Storage administration pages:

- By opening a WebUser Session, and using the Files link in the WebUser Interface navigation panel.
- By opening the `Index.wssp` file in their own personal Web site:
  `http://domain.dom:8100/~username/Index.wssp`
  A browser will present a Login (authentication) dialog box, and the users should enter their Account names and passwords in order to open the Web site administration page.

Server administrators with the All Domains access right and Domain administrators with the `CanAccessWebSites` access right can access File Storage in other Accounts.

Server and Domain administrators can access File Storage of any Account using the WebAdmin Interface: the Account management pages have the Files link in their navigation panels.

All management methods use similar HTML pages for File Storage administration, see the WebUser Interface Files section for the details.

# HTTP-based Management

File Storage data can be modified using the HTTP 1.1 `PUT`, `DELETE`, and `MOVE` methods. Some HTML design tools can use these methods to upload files to the server.

These HTTP requests should contain the Authentication information: the Account name of the File Storage owner or the Account name of a Server/Domain Administrator, and the password for that Account.

# HTTP Access to File Storage

CommuniGate Pro allows each user to be presented on the World Wide Web with a personal Web site. The URL for the *accountname@domainname* Account File Storage is:
`<http://domainname:port/~accountname>`   where the *port* is the WebUser port.
For example, the `jsmith@client1.com` account has a personal Web site at:
`<http://client1.com:8100/~jsmith>`

Personal Web sites use the same HTTP port as the WebUser Interface (the port 8100 by default).

In addition to the `~` prefix, an alternative prefix can be specified in the Domain Settings. The alternative prefix can be an empty string.

All Routing Rules discussed in the Access section apply to the personal Web site URLs, so Account and Domain aliases can be used in the personal Web site URLs.

Personal Web sites can be accessed without a prefix, using just the server part of the URL string. When the CommuniGate Pro server receives an HTTP connection on the its WebUser port, it uses the special Domain Routing procedure.

If the domain name `user.domain.com` has a DNS A-record pointing to the IP address of the CommuniGate Pro server, and the CommuniGate Pro Router has the following record:

```
<LoginPage@user.domain.com> = userA@domainB.com
```
and the Account userA exists in the CommuniGate Pro Domain domainB, then the URL `http://user.domain.com/` can be used to access the personal Web site (File Storage) of the userA@domainB.com Account.

File Storage must not contain any `index.wssp` file. This name is reserved for the File Storage Management forms.

The home (default) page of a personal Web Site should have the `default.html` name. This means that when the file name is not specified explicitly, the `default.html` name is assumed. If a File Storage has folders (subdirectories), then the request with the `http://server:port/prefix user/folder/` URL retrieves the `default.html` file from that subdirectory.

The name of the default page is specified as an [Account Setting](#) and it can be modified on the per-Account basis.

---

# FTP Access and Management

File Storage data can be accessed, modified, and managed using the CommuniGate Pro [FTP module](#). When an Account user connects to the FTP module, the FTP "root directory" as well as the "current directory" are set to the Account File Storage top directory.

---

# WebDAV Access and Management

File Storage data can be accessed, modified, and managed using the CommuniGate Pro [HTTP module](#) WebDAV extension.
use `http://server:port/WebDAV/` or `https://server:port/WebDAV/` URLs to configure a WebDAV client.

Access to the /WebDAV/ realm requires authentication, and the authenticated Account and its Domain must have the [WebSite Service](#) enabled.
This realm presents the authenticated Account File Storage top directory.

The File Access WebDAV protocol works over the [HTTP](#) protocol, using the HTTP User Module. Open the HTTP User Module settings, and find the Sub-Protocols panel:

| Sub-Protocols |
|:---:|
| FileDAV:   All Info |

Use the FileDAV Log setting to specify the type of information the File Access WebDAV module should put in the Server Log.

The File Access WebDAV module records in the System Log are marked with the `FileDAV` tag.

---

# Foreign File Access

The CommuniGate Pro allows an Account user to access File Storage in other Accounts.
Access to these foreign Files (also called shared Files) is controlled via the [File Access Rights](#).

To access a file or a file directory in a different Account, the file name should be specified as `~accountname/filename`. For example, to access the `images/pict01.jpg` file in the Boss Account, the file name should be specified as `~Boss/images/pict01.jpg`.

If there are several local Domains on the Server, files in a different Domain can be accessed by specifying full Account names. To access the `images/pict01.jpg` file in the Account `designer` in the client.com Domain, the file name should be specified as `~designer@client.com/images/pict01.jpg`.

Account names specified after the "~" sign are processed with the Router, so Account Alias names can be used instead of the real Account names, and all Routing Table rules are applied.

# File Subscription

Each Account has a *file subscription* set - a set of file and/or file directory names.

This list is maintained with various clients. Usually, it contains the names of foreign file directories, such as `~accountName/dir1/dir2/`, letting clients show some preselected foreign file directories.

| Introduction |
|---|
| Installation |
| SysAdmin |
| Network |
| Objects |
| **Storage** |
| E-mail |
| Real-Time |
| Access |
| Services |
| Directory |
| Clusters |
| Applications |
| Miscellaneous |
| Licensing |
| |
| WebMail |
| Pronto! |
| PBX |

| Storage | Mailboxes | Files | **External** |
|---|---|---|---|

# External Storage

- **Configuring External Storage**
- **Remote Access via IMAP**
- **Synchronized Storage via AirSync**

CommuniGate Pro Accounts can utilize mailbox storage of other (external) systems. There are 2 main methods for mailbox storage integration:

- Remote Access. The mailboxes stored on a remote system are seen as CommuniGate Pro Account own mailboxes. When such a Mailbox is opened, CommuniGate Pro connects to the external system and performs all requested Mailbox operations remotely, reading and modifying data directly in the external system mailbox storage.
- Synchronized Storage. Mailboxes are stored locally, within the CommuniGate Pro Accounts. The CommuniGate Pro periodically connects to the external system and synchronizes its local storage with the external system mailbox storage: messages added to or deleted from the CommuniGate Pro Mailboxes are added to and deleted from the external storage mailboxes, and messages added to or deleted from the external storage mailboxes are added to and deleted from the CommuniGate Pro mailboxes.

# Configuring External Storage

To configure External Mailbox Storage for an Account, open the Account Settings using the WebAdmin Interface, and open the Mail Settings page in the Mail section:

| External Storage | | |
|---|---|---|
| External Storage Mode: IMAP | External Server: | imap.service.dom:993 |
| Account Name: ^0 | Password: | ^0 |

`External Storage Mode`
     The External Storage integration method.

`External Server`
     The name of the external mailbox storage system. It can contain a macro symbol `^0` which is substituted with the CommuniGate Pro Account Domain name.
     The name may contain an IP port suffix, and it may include the `tls:` prefix to indicate that CommuniGate Pro must connect to that system via a secure (encrypted) connection.
     If the specified IP port is a standard secure port for the selected protocol (993 for IMAP, 443 for HTTP-based protocols), secure connects are used with our without the `tls:` prefix specified.

`Account Name`

The external mailbox storage system credential.

This settings can contain a macro symbol `^0` which is substituted with the CommuniGate Pro Account name.

`Password`
The external mailbox storage system credential.

This settings can contain a macro symbol `^0` which is substituted with the CommuniGate Pro Account password. The CommuniGate Pro Password should not be stored using a one-way hash method, otherwise a synchronization attemp will fail.

# Remote Access via IMAP

When the External Storage Mode setting is set to `IMAP`, the Account Mailbox storage is extended using the external mailbox storage system, via IMAP connections:

- All external mailbox storage mailboxes are visible in the mailbox list. If there is a local Mailbox with the same name as an mailbox in external mailbox storage, the external mailbox is used.
- When a mailbox is being opened, a local Mailbox with the specified name is opened, if present. If not present, an IMAP connection to the external mailbox storage is created. It is used to open the specified external mailbox and to process mailbox operation by reading and modifying the external mailbox.
- When there is a request to create a mailbox, it is created locally, inside the CommuniGate Pro Account, if the requested Mailbox Class is not "Email" (empty), or if the parent mailbox is a Local Mailbox. In all other cases the mailbox is created in the external mailbox storage.

# Synchronized Storage via AirSync

When the External Storage Mode setting is set to `AirSync`, the Account Mailbox storage is synchronized with the external mailbox storage system using the AirSync protocol:

- The mailbox tree is synchronized every time a mailbox listing operation is applied to the Account
- Mailbox creation, renaming, deletion operations are applied to the external mailbox storage first, and then the mailbox trees are synchronized.
- Each Mailbox is synchronized when it is being opened and then periodically while the Mailbox is kept open.
- When a message or other item is being moved from one Mailbox to a different Mailbox, this operation is applied to the external mailbox storage, and then both Mailboxes are synchronized.

| Overview | Rules | Filters | SMTP | Local | RPOP | LIST | PIPE |
|----------|-------|---------|------|-------|------|------|------|

# E-Mail Transfer

- **E-mail Sources and Destinations**
- **Submitting Messages**
- **Queue Limits and Foldering**
- **Routing**
- **Enqueueing**
- **Delays and Suspensions**
- **Dequeueing**
- **Monitoring a Queue**
- **Monitoring a Message**

One of the main functions of the CommuniGate Pro Server is E-mail Message transfer. Acting as an MTA (Mail Transfer Agent), the Server accepts messages from various sources (modules, internal components, etc.), and delivers (transfers) them to remote or local destinations using the same or different modules.

While all submitted messages are stored as individual files in the Queue directory inside the CommuniGate Pro "base directory", each message can be enqueued into several different queues (if it has several recipients). Each communication Module can maintain one or several logical queues. For example, the SMTP Module maintains one queue for each Internet domain.

# E-mail Sources and Destinations

The CommuniGate Pro Server has the following set of message sources:

- The SMTP Module submits messages received from mailer applications and from other mail servers via the Internet. The Submit Address tag used: SMTP.
- The RPOP Module submits messages retrieved from remote POP servers. The Submit Address tag used: RPOP.
- The Local Delivery Module submits messages generated with the Account-level (Account and Domain) Automated Mail Processing Rules. The Submit Address tag used: RULE.
- The Local Delivery Module re-submits messages directed to 'all' addresses and redirected to all Forwarders in the specified Domain. The Submit Address tag used: ALLF.
- The PIPE Module submits messages received from external applications via interprocess communication channels, and the messages generated and stored in the special Submitted directory. The Submit Address tag used: PIPE.
- The POP Module submits messages received from certain mailer applications employing the XTND XMIT protocol extension. The Submit Address tag used: POP.
- The IMAP Module submits messages received from MAPI client applications. The Submit Address tag used: IMAP.
- The WebUser Interface Module submits messages composed within Web browsers. The Submit Address tag used: HTTP.

- The [XIMSS](#) Module submits messages composed within XIMSS clients. The Submit Address tag used: `XIMSS`.
- The [Real-Time Applications](#) submit messages such as incoming voicemails. The Submit Address tag used: `PBX`.
- The [Enqueuer](#) component submits messages generated with the Server-wide and Cluster-wide [Automated Mail Processing Rules](#). The Submit Address tag used: `SRULE`.
- The [Dequeuer](#) component generates and submits delivery notification messages. The Submit Address tag used: `DSN`.
- The [WebUser Interface](#) and [XIMSS](#) modules generate and submit Message Disposition Notification ("Read Receipts") messages. The Submit Address tag used: `MDN`.
- The [Calendaring](#) component generates and submits Event Request Reply messages. The Submit Address tag used: `ICAL`.
- The [LIST](#) Module submits messages to be distributed to mailing list subscribers. The Submit Address tag used: `LIST`.
- The [LIST](#) Module submits reply messages generated when processing administration requests. The Submit Address tag used: `LSRV`.
- The [LIST](#) Module submits messages directed to individual subscribers (Warnings, Hello, Bye, etc.) The Submit Address tag used: `LSTM`.
- The [LIST](#) Module submits messages directed to the list owner. The Submit Address tag used: `LSTO`.
- The [LIST](#) Module re-submits messages directed a [Group](#). The Submit Address tag used: `GROUP`.
- The [Triggers](#) component generates and submits notification messages. The Submit Address tag used: `EVENT`.
- The [Lawful Interception](#) component generates and submits report messages. The Submit Address tag used: `LWIT`.

The CommuniGate Pro Server transfers messages to the following destinations:

- the [SMTP Module](#) transfers messages to other SMTP mail servers via the Internet;
- the [LIST Module](#) accepts and processes messages with mailing list postings, and with various list administration requests;
- the [Local Delivery Module](#) transfers messages to local user mailboxes;
- the [PIPE Module](#) transfers messages to external applications via interprocess communication channels;
- the Queue Component itself transfers (discards) messages routed to NULL or ERROR addresses;

The following diagram illustrates the message flow inside the CommuniGate Pro Server:
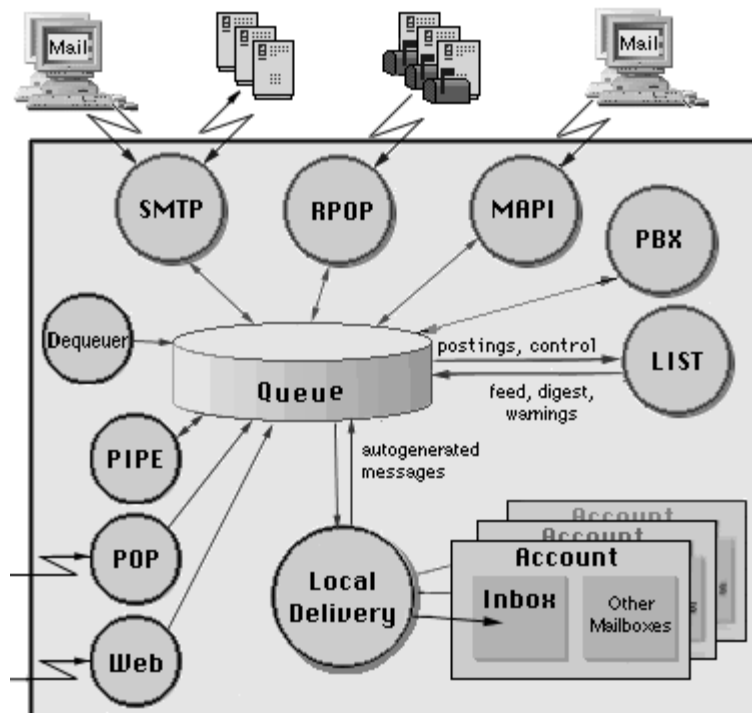
# Submitting Messages

All messages are created as temporary files. They are stored in the `Queue` directory as files with the `.tmp` extension. A Module or an internal component stores the message envelope and the message itself in such a file and then submits it to the Enqueuer component for processing.

The message envelope is a set of text lines. Each line specifies either the message return-path, or one message recipient address, or message delivery options.

If a Module fails to compose a message (for example, an SMTP connection breaks during message transfer), the Module discards of the temporary file.

When a message is completely composed and submitted to the Enqueuer, the file extension is changed to `.msg` and the message is scheduled for processing.

When the Server restarts, the Enqueuer component finds all files with the `.msg` extension stored in the `Queue` directory and resubmits them for processing.

Open the General pages in the Settings realm of the CommuniGate Pro WebAdmin Interface, and find the Temp File Manager panel panel on the Others page:

| Temp File Manager | |
| --- | --- |
| Log Level: | Failures |
| Recycle Temp Files: | Enabled |

`Log`

This setting specifies what kind of information the Temporary Files manager should put in the Server Log. Usually you should use the `Failures` (file system error reports) level. But when you experience a problem with some Module submitting messages, you may want to set this setting to `Low-Level` or `All Info`: in this case file i/o operations will be recorded in the System Log as well. When the problem is solved, set the TempFiler Log setting to its regular value, otherwise your System Log files will grow in size very quickly.

The Temporary Files manager Log records are marked with the `TEMPFILE` tag.

`Recycle Temp Files`

Enable this option to improve performance of your system under heavy load: discarded temporary file are not deleted, and they are reused.

# Queue Limits and Foldering

If a Server is heavily loaded, it can contain thousands of message files in its Queue directory. Many operating and file systems cannot efficiently process large directories. You may want to split the Queue directory into several subdirectories, each containing a portion of Temporary and Queue files.

You may want to limit the total number of messages in the CommuniGate Pro Queue. When this number is reached, the modules reject attempts to submit new messages.

Use the WebAdmin Interface to specify the Queue processing options. Open the Mail pages in the Settings realm, then open the Queue page:

| Message Queue | |
|---|---|
| Log Level: | Low Level |
| Queue Size Limit: Unlimited | Queue Foldering: Auto |

Queue Limit
> If you specify a limit with this option, the CommuniGate Pro modules (SMTP, PIPE, RPOP, MAPI, WebUser) will stop accepting new messages when the number of messages in the Queue exceeds the specified limit.

Queue Foldering
> This setting specifies where the CommuniGate Pro Server will create new Temporary files, and, as a result, how the message files will be stored in the Queue Directory.
> If you select the 0 (zero) value, no subdirectory will be created in the Queue directory, and all Temporary files will be created directly in the Queue directory.
> If you select the 10 or 100 value, subdirectory with NN names (where NN is a 2-digit number from 00 to 99) will be created in the Queue directory. The newly created Temporary files will be created inside these subdirectories. The server will use 10 or 100 subdirectories, depending on this setting value.
> If you select the Auto value, Queue subdirectories will be used when the total number of messages in the Queue exceeds 5000.

# Routing

When a message is submitted for processing, the Enqueuer component examines its envelope information. Each recipient address is parsed and passed to the [Router](#) component. The Router component decides which Module or component should process each recipient address.

# Enqueueing

When all recipient addresses are parsed and routed, the Enqueuer component applies the [Server-Wide Rules](#) to the message. Then it passes the message to the modules specified with the Router component.

Communication modules do not process E-mail messages immediately, but enqueue them into the module-specific queues. The SMTP Module creates and maintains a queue for each Internet domain, the Local Delivery Module creates and maintains a queue for each local Account, etc.

The Enqueuer component can enqueue messages:

- synchronously: as soon as messages are composed and submitted, Routing and Sever-wide/Cluster-wide Rules are applied, and the message composer component is informed if the message has been rejected (for example, if an External Filter has found a virus in the message). For example, the SMTP incoming channel sends a negative response to the sender of an infected message, the message is not enqueued, and no error report (bounce) messages are generated for infected messages.
- asynchronously: the composer component immediately receives a positive response, and it can continue processing without waiting for the Enqueuer component. For example, the SMTP incoming channel can immediately start receiving the next message.

Most of the internal components enqueue messages asynchronously, as they cannot do anything useful if a message is rejected with the Enqueuer component. The components receiving messages directly from users or remote systems (SMTP, MAPI, WebMail, XIMSS) try to enqueue messages synchronously, so if a message is rejected with the Enqueuer component, the submitting agent (a user or a remote system) can get an error message.

Use the WebAdmin Interface to configure the Enqueuer component. Open the Queue page in the Settings realm.

| Message Enqueuer | | | |
|---|---|---|---|
| Log: | Low Level | Processors: | 1 |
| Hop Counter Limit: | 20 | Enqueue Asynchronously if Senders are: | anybody |

Enqueuer Log
> Use this setting to specify what kind of information the Enqueuer component should put in the Server Log.
> Usually you should use the `Failures` (file system error reports) level.
> The Enqueuer component Log records are marked with the `ENQUEUER` tag.

Enqueue Asynchronously
> Use this option to specify when the asynchronous Enqueuer mode should be used:

> anybody
>> always use the asynchronous mode

> nobody
>> never - always use the synchronous mode

> authenticated
>> use only for messages received from authenticated sources (SMTP AUTH, AirSync, MAPI, WebMail, XIMSS, etc.)

> unauthenticated
>> use only for messages not received from authenticated sources

> clients
>> use only for messages received from authenticated sources or from the Network IP Addresses included into the Client IP Addresses list.

> non-clients
>> use only for messages received from neither from an authenticated source, nor from a Network IP Address included into the Client IP Addresses list.

Processors
> Use this setting to specify the number of Enqueuer processors (threads).
> If the asynchronous Enqueuer mode is not used, the Enqueuer threads are needed only to enqueue existing messages after the Server restart, and to enqueue messages generated with the internal Server component, so you may want to use 2-3 threads only.
> If the asynchronous Enqueuer mode is used, then each message is processed with some Enqueuer thread. In this case, 10-20 Enqueuer threads should be enough even for a heavily loaded Server.

> You should increase the number of Enqueuer processors if:

> - there are very many Server-Wide Rules;
> - Server-Wide Rules use the `Execute` action to start external programs;
> - one or more External Filter (anti-virus, content filtering, etc.) programs are enabled;
> - incoming load is very high, with many messages being submitted every second via the SMTP or other modules.

The `numEnqueuerMessages` Statistics element shows the number of messages that have been received, but not

enqueued yet. If this number is growing, you need to increase the number of Enqueuer processors.

Hop Counter Limit
>When a message is being received by any host or Module, it gets an additional Received: header field. The Hop Counter is the number of Received: header fields in the message. If a message contains too many Received: header fields, it may indicate that a message is in some sort of mail loop. This parameter specifies the Limit for the Hop Counter - any message that has more Received: header fields than specified in this setting is rejected by the ENQUEUER - without any attempt to deliver it to the recipients specified in the message envelope.

# Delays and Suspensions

When a communication Module fails to transfer a message, it uses the kernel queue management component to delay processing.

- a Module can delay an entire queue: for example, the SMTP Module can delay a queue created for an Internet domain, if it cannot connect to that domain or its relays;
- a Module can delay an individual queued message: for example, the SMTP Module can delay a message if the receiving host rejects this particular message (transition failure);
- a Module can delay an individual recipient address in a queued message: for example, the SMTP Module can delay an address if the receiving host rejects that particular address (transition failure).

# Dequeueing

When a communication Module transfers a message or when it rejects a message because of a fatal error, it removes the message from the Module queue. The Module composes a delivery report and passes it to the Dequeuer component.

The Dequeuer component processes delivery information. If requested, it composes Delivery Status Notification (DSN) messages and submits them back to the system for delivery to the original message sender. When a message has several recipients, the Dequeuer component may choose to delay DSN generation, so each DSN message can contain reports about several recipients.

When all message recipients are processed and the message is dequeued from all queues, the Dequeuer component removes the message file from the Queue directory.

Use the WebAdmin Interface to configure the Dequeuer component. Open the Queue page in the Settings realm.

| Message Dequeuer | | | |
|---|---|---|---|
| Log: | Major & Failures | Processors: | 1 |
| Reporting Delay: | 15 seconds | Send Return-Receipts to: | everybody |
| On Failure, Return: | body by default | Log Message Delivery: | Disabled |
| Copy Failure Reports to: | | Allow Message Revocation: | Disabled |

Dequeuer Log
>Use this setting to specify what kind of information the Dequeuer component should put in the Server Log. Usually you should use the Major & Failures (delivery reports) level.
>The Dequeuer component Log records are marked with the DEQUEUER tag.

Processors
> Use this setting to specify the number of Dequeuer processors (threads). Usually one Dequeuer thread is enough even for a heavy-loaded server. Only if your Server performs some kind of special message processing and has to generate a lot of DSN messages, should you use several Dequeuer threads.

Reporting Delay
> Use this setting to specify the maximum delay between the moment when a message was transferred or failed and the moment when a delivery report is generated. The more the delay, the more reports can be placed in one DSN message. A DSN message is generated immediately after the last message recipient is processed.

On Failure, Return
> Use this setting to specify what portion of a failed message should be included into the DSN (error report) message.

> - If the sender has not specified this option explicitly, and the `headers by default` option is selected, only the failed message headers will be returned;
> - If the sender has not specified this option explicitly, and the `body by default` option is selected, the entire failed message will be returned;
> - If the `always headers` option is selected, only the message headers are included into the DSN message, even if the message sender has specified that the entire message should be returned on failure;
> - If the `always body` option is selected, the entire message is included into the DSN message, even if the message sender has specified that only the message headers should be returned on failure.

Copy Failure Reports
> When this option is enabled, all error messages generated with the CommuniGate Pro Dequeuer are sent to both the failed message return-path and to the specified E-mail address.

Send Return-Receipts to
> Positive reports (delivery reports and relay reports) are generated only if the message sender has requested them. Use this setting to specify who can request these reports:
> | | |
> |---|---|
> | everybody | reports are generated and sent whenever the message sender requests them. |
> | clients | reports are generated and sent if the sender has submitted the message from a Client IP Address or if the message has been submitted by an authenticated user. |
> | authenticated | reports are generated and sent if the message has been submitted by an authenticated user (via WebUser, XIMSS, SMTP AUTH, MAPI, etc.) |
> | nobody | positive reports are not generated. |

Log Message Delivery
> If this option is enabled, then the Dequeuer places a record for each E-mail delivery into the EMails supplementary Log.

Allow Message Revocation
> If this option is enabled, then the Server processes messages with `X-Special-Delivery: recall` header and tries to cancel delivery of the messages that were recalled.

---

# Monitoring a Queue

Transfer modules (such as SMTP, Local Delivery, LIST, and PIPE) maintain one or several queues for messages to be delivered. Each Module uses its own methods to build the queues (for example, the SMTP Module usually builds a separate queue for each remote domain to deliver to, while the Local Delivery Module builds a separate queue for each local Account), see the manual section describing the Module for more details.

To open a Module queue, click the queue name link on Module Monitor page. The Module ("host") queue page opens:

| Module | Status | Last Problem |
|---|---|---|
| SMTP | Processing | no response |

| Message | In Queue | Return Path | Recipients | Size | Delayed | Last Problem |
|---|---|---|---|---|---|---|
| 44220010 | 5 min | <system_admin@domain.dom> | 2 | 634 | | |
| 44220003 | 15 min | <user2@domain3.dom> | 1 | 26K | | |

The table header contains the information about the entire queue ("host"):

Module
>    The link to the Monitor page of the Transfer Module this queue belongs to.

Status
>    `Active` - this queue is being processes by the Module.
>    `Ready` - this queue can be processes by the Module at any time.
>    `Delayed till` *time* - this queue will not be processes by the Module until the time specified. The queue can be delayed because there was a queue-wide transfer operation error (an SMTP host did not respond, or a Local Account is over its quota, etc.)

Last Error
>    This field indicates the last queue-wide transfer operation error.

The table contains a record for each message in the queue. For each enqueued message, the following information fields are displayed:

Message
>    The message internal ID. You can use this link to open the Message Monitor page.

In Queue
>    The time the message has spent in this queue.

Return Path
>    The message envelope "Return-Path" address.

Recipients
>    The number of message addresses that should be delivered to the "host" this queue is created for. For example, a message directed to user1@company.dom and user2@company.dom addresses via SMTP will appear once in the company.dom SMTP queue, with the indicated number of addresses being 2.

Size
>    The message size.

Delayed and Last Error
>    If delivery failed for this particular message, the Transfer Module could delay this message individually (rather than delaying the entire queue). In this case, this field will show the time when the Module will try to deliver the message again, and the Last Error field will show the information about the cause of the transfer operation failure.

If you have the Can Release Queues access right, and the Queue has the Delayed status, the Host Queue Monitor page contains the Release Now button. Click this button to remove the delay interval set for this queue and to allow the Module to process the queue immediately.

If you have the Can Reject Queues access right, and the Queue has the Delayed or Ready status, the Host Queue Monitor page contains the Reject Host Queue button. Click this button to reject the queue:

> Suppress Failed Delivery Reports

The specified text is used to generate DSN messages (error reports) for all messages placed into this queue.

If the Suppress Failed Delivery Reports option is selected, no DSN message is generated when the queue messages are

being rejected.

---

# Monitoring a Message

To monitor the status of a message in the Server Queue, click the message link on the Module queue or other Monitor page. The Message Monitor page opens:

| | |
|---:|:---|
| **Sender:** | user@mail.communigate.com |
| **Return-Path:** | <user@mail.communigate.com> |
| **Message-ID:** | <list-26325892@mail.communigate.com> |
| **Subject:** | Re: FormMail.pl configured for CGP |
| **Size:** | 37222 bytes (34018 in envelope) |
| **Submitted:** | Sat, 03 May 2014 09:16:58 -0800 (12hours ago) |

| Recipient | Module | Queue | Status | Retry in |
|---|---|---|---|---|
| postmaster@domain1.dom | SMTP | domain1.dom | Delayed | 24min |
| john.smith@domain2.dom | SMTP | domain2.dom | Delayed | 57min |

The first part of the page shows the message attributes: the envelope Return-Path, Message-ID, Envelope-ID (if present), message Subject, and the time the message was submitted to the Server Queue.

If the message has been submitted by an authenticated user, the Sender line displays the sender's Account name.

The second part of the page lists all active message recipient addresses (if a message address has been already processed, it is not shown). For each address the following information is displayed:

Recipient
> Recipient address.

Module
> The name of the Module that will process this address. This is also a link to the Module Monitor page.

Queue
> The name of the Module queue containing this address. This is also a link to the Module queue Monitor page.

Status
> `Processing` - the Module is processing this address.
> `Ready` - the Module can start processing of this address at any time.
> `Suspended` - the Module has delayed processing of this message.
> `Delayed` - the Module has delayed processing of the entire queue containing this address.

Retry In
> Time till the Module resumes processing of this message or the queue containing this message.

If you have the Can Reject Queues access right, the Message Monitor page contains the Reject Message button. Click this button to reject all active message addresses.

If the message has been submitted by an authenticated sender, click the Reject All Sender's Messages button to reject all messages submitted by this sender.

Suppress Failed Delivery Reports

Only the addresses that are not being processed can be rejected.

The specified text is used to generate DSN messages (error reports) for all rejected recipient addresses.

If the Suppress Failed Delivery Reports option is selected, no DSN message is generated when the message addresses are being rejected.

If you have the Can View Queue Messages access right, the Message Monitor page contains the Message content (it is displayed using the Unnamed Stock Skin).

| | |
|---|---|
| **From:** | "User Name" <userName@server.dom> |
| **Subject:** | Some subject here |
| **Date:** | Fri, 07 Dec 2007 02:56:41 -0800 |
| **Message-Id:** | <web-12490003@server.dom> |
| **X-Mailer:** | CommuniGate Pro WebUser v5.3.5 |

```
TEST - TEST TEST

--
This is a test letter with one attachment.
```

**Attachment:** TestDocument.doc (29K)

# Automated E-mail Processing Rules

- **Specifying Message Rules**
- **Rule Conditions**
- **Rule Actions**
- **Macro Substitution**
- **Vacation Message**
- **Copy All Mail Simplified Rule**
- **Junk Processing Simplified Rules**
- **Logging Rules Activity**

The CommuniGate Pro Server can automatically process messages using sets of Automated Rules.

This section describes the Automated Processing Rules for E-mail messages, also called the Queue Rules.

The Server-Wide and Cluster-wide Rules are applied to all messages submitted to the Server and to the Cluster. These Rules are applied by the Enqueuer component, before it enqueues a message into the transfer module queues.

When a message is directed to an Account on this CommuniGate Pro Server, the Local Delivery module applies the Account-level Rules. The Account-level Rules are the Rules specified for the particular Account along with the Rules specified for the Account Domain.

## Specifying Message Rules

System administrators can specify Server-Wide and Cluster-wide Message Rules. Use the WebAdmin Interface to open the Mail pages in the Settings realm, and click the Rules link.

System administrators can specify Account Rules using links on the Account Settings page.

Account users can specify their Rules themselves, using the WebUser Interface. System or Domain administrators can limit the set of Rule actions a user is allowed to specify.

System and Domain Administrators can specify Domain-Wide Rules using the Rules links on the Domain Settings page.

See the Automated Rules section to learn how to set the Rules.

## Rule Conditions

Each Rule can use universal conditions specified in the [Generic Rules](#) section.
This section describes the additional Rule conditions you can use in E-mail (Queue) Rules.


**From** [is | is not | in | not in] *string*
**Sender** [is | is not | in | not in] *string*
      This condition checks the message `From` or `Sender` address.
      If a message has no From/Sender address, the condition is met if its operation is `is not` or `not in`.

      Sample:

> | From | is |
> | --- | --- |

      This condition will be met for all messages coming from any account in any of the communigate.com subdomains.


      The same as above, but the message Sender, Reply-To, To, or Cc address is checked.

**To** [is | is not | in | not in] *string*
**Cc** [is | is not | in | not in] *string*
**Reply-To** [is | is not | in | not in] *string*
      The message Reply-To, To, or Cc address is checked.

      If a message has several addresses of the given type, the condition is met if it is true for at least one address. If a message has no addresses of the specified type, the condition is not met.


**Any To or Cc** [is | is not | in | not in] *string*
      The same as above, but all message To AND Cc addresses are checked. If the message has no To/Cc addresses, the condition is not met.

**Each To or Cc** [is | is not | in | not in] *string*
      All message To AND Cc addresses are checked. The condition is met if it is true for each To and Cc address of the message, or if the message has no To/Cc addresses.

      Sample:

> | Each To or Cc | in |
> | --- | --- |

      This condition will be met for messages where all To and CC addresses are addresses in the mycompany.com domain or addresses in the mydept.mycompany.com domain.


**Return-Path** [is | is not | in | not in] *string*
      This condition compares the message "Return-Path" (a.k.a. MAIL FROM) envelope address with the specified string.

**'From' Name** [is | is not | in | not in] *string*
      The same as above, but the instead of the address, the "address comment" (the real name) included in the From address is checked.

      Sample:

> | 'From' Name | is |
> | --- | --- |
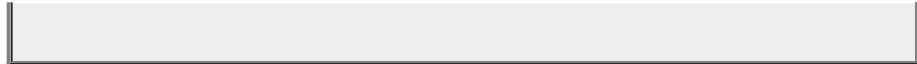
This condition will be met for messages with the following `From:` addresses:

```
From: jsmith@company.com (John J. Smith)
From: "Bill J. Smith" b.smith@othercompany.com
From: Susan J. Smith <susan@thirdcompany.com>
```

**Subject** [is | is not | in | not in] *string*

This condition checks if the message subject is (or is not) equal to the specified string.

Sample:



This condition will be met for messages with the following Subject fields:

```
Subject: we urgently need your assistance
Subject: Urgent!
```

**Message-ID** [is | is not | in | not in] *string*

This condition checks if the message ID is (or is not) equal to the specified string.

Sample:
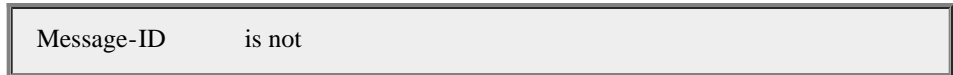


This condition will be met for all messages without the Message-ID flag and for messages that have Message-ID without the `@` symbol.

**Message Size** [is | is not | less than | greater than] *number*

This condition checks if the message size is less than (or greater than) the specified number of bytes.

Sample:
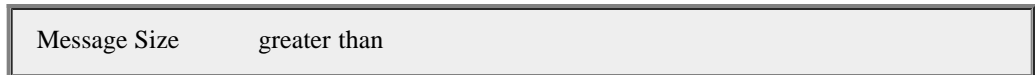


This condition will be met for messages larger than 100 kilobytes.

**Human Generated**

This condition checks if the message is not generated by some automatic message generating software.
**Note:** this condition has no parameters, so the operation code and the parameter value (if any) are ignored.

It actually checks that the message header does not contain any of the following fields: `Precedence: bulk`, `Precedence: junk`, `Precedence: list`, `X-List*`, `X-Mirror*`, `X-Auto*` (except for `X-Auto-Response-Suppress`), `X-Mailing-List`, `Auto-*`
This condition also checks that the message has a non-empty Return-Path.

**Header Field** [is | is not | in | not in] *string*

This condition checks if the message RFC822 header contains (or does not contain) the specified header field. The fields added using the `Add Header` operation (see below) are checked, too.

Sample:

| Header Field | is not | |
|---|---|---|

**Any Recipient** [is | is not | in | not in] *string*

This condition compares message "Envelope" addresses and the specified *string*. If this condition is used in an Account-Level Rule, only the addresses routed to that account are checked.

The addresses are processed in the form they had *before* the Router Table and other routing methods have modified them. If an account has several aliases, this condition allows you to check if a message was sent to a specific account alias.

Messages can be submitted to the server using the ESMTP ORCPT parameter. This parameter specifies how the address was composed on the sending server, before the relaying/forwarding server has converted it to a different address. In this (rare) case, that server can use the ESMTP ORCPT parameter to specify the original address.

Sample:

- a message was composed somewhere and sent to the address user1@domain1.com;
- the domain1.com server received the message and converted that envelope address to user2@domain2.com (mail forwarding);
- the domain1.com server relayed the message to your CommuniGate Pro server domain2.com;
- the domain2.com CommuniGate Pro server received a message;
- the domain2.com CommuniGate Pro server found that the user2 is an alias of the user3 account, and the server routed the message to that user3 account.

If the domain1.com server is an advanced server and informed the domain2.com CommuniGate Pro server that the original address was user1@domain1.com, the string <user1@domain1.com> is used when the Recipient condition is checked.

If the domain1.com server has not informed your server about the original address, the <user2@domain2.com> string is used when the Recipient condition is checked.

The condition is met if it is met for at least one envelope address.

**Each Recipient** [is | is not | in | not in] *string*

The same as above, but the condition is met only if it is met for all message envelope addresses (if used in an Account-Level Rule - for all message addresses routed to that account).

**Source** [is | is not | in | not in] *string*

This condition checks the source where from the message was received:

| | |
|---|---|
| trusted | via SMTP from a computer with the network address listed in the Client IP Addresses list |
| whitelisted | via SMTP from a computer with the network address listed in the White Hole Addresses list |

| `authenticated` | via SMTP, WebUser, MAPI, POP XMIT, Rules - when the originator of the message has been authenticated |

Sample:

| Source | not in | |
|---|---|---|

**`Security`** [is | is not | in | not in] *string*

This condition checks if the message is an encrypted or a signed one. It compares the following string with the condition operand:

| `SMIME:encrypted` | if the message is encrypted using the S/MIME standard |
| `SMIME:signed` | if the message is digitally signed using the binary S/MIME standard (PKCS7) |
| `signed` | if the message is digitally signed |
| | (*empty string*) in all other cases |

Sample:

| Security | is not | |
|---|---|---|

**`Calendar Method`** [is | is not | in | not in] *string*

This condition checks if the message contains a calendar part of particular type (`REQUEST`, `REPLY`, `CANCEL`). This condition can be used in Domain-wide or Account-level rules only.

Sample:

| Calendar Method | not in | |
|---|---|---|

The following conditions can be used in Server-Wide Rules only:

**`Any Route`** [is | is not | in | not in] *string*

This condition checks a message "Envelope Recipient" address - the address actually telling the Server were to transfer the message to. The condition compares the routing information string for a recipient address and the specified *string*.

The condition is met if it is met for at least one envelope recipient address.

The message address routing information is presented in the following format:

> *module*(*queue*)*address*

where *module* is the name of the module the address is routed to, *queue* is the name of the module queue the

address is routed to, and *address* is the address in that queue.

For example, the envelope recipient `user@domain` address can be routed to:

| | |
|---|---|
| `SMTP(domain)user@domain` | if `domain` is a remote domain |
| `LOCAL(user)` | if `domain` is the Main Domain |
| `LOCAL(user@domain)` | if `domain` is a secondary CommuniGate Pro Domain |

If you plan to use this type of Rule condition, use the Test button on the WebAdmin Interface Router page to see how various addresses are routed.

**Each Route** `[is | is not | in | not in]` *string*
> The same as above, but the condition is met only if it is met for all message envelope addresses.

---

# Rule Actions

Each Rule can have zero, one, or several actions. If a message meets all the Rule conditions, the Rule actions are performed.

You can use all universal actions described in the [Generic Rules](#) section. This section describes the additional Rule actions you can use in E-mail (Queue) Rules.

`Stop Processing`
> This action should be the last one in a Rule. Execution of this Rule stops and no other (lower-priority) Rules are checked for that message. The message is stored in the INBOX.

`Discard`
> This action should be the last one in a Rule. Execution of this Rule stops and no other (lower-priority) Rules are checked for that message.
> The message is not stored in the INBOX, but a positive Delivery Notification message is sent back to the message sender (if requested).
>
> Sample:
> ```
> IF From is *that_annoying_guy@*
> THEN
> Discard
> ```

`Reject With` *[error message text]*
> See the [Rules](#) section.
> If the action parameter text is not empty, it is used as the error message text.
> You can still store the rejected message using the Store action before the Reject action.
>
> Sample:
> ```
> IF Subject is *UCE*
> THEN
> Reject   please do not send such messages here
> ```

`Mark` *flagName [, flagName...]*
> This action sets or resets the specified message [flag(s)](#).
> Initially the set of message flags contains:
>
> - the `Media` flag - if the message contains a voicemail or videomail (if the message has the `audio/*`,

`video/*`, or `multipart/voice-message` Content-Type).

- the `Hidden` flag - if the message header contains the Sensitivity field with the `private` value.

Flag [Names](#) can be used to add flags to the set, while Flag [Negative Names](#) can be used to remove flags from the set.

When the message is stored in a Mailbox as a result of the `Store in` action, as well as when the message is stored in the INBOX after all Rules are applied, the message is stored with the specified flag set.

Sample:
```
IF Sender is *list*
THEN
Mark Flagged,Read
```

`Add Header` *header fields*

This action adds RFC822 header fields to the message. Initially, the set of additional message header field contains the Return-Path field generated using the return-path in the message envelope.

When a message is stored, sent, copied, or sent to an external program, the additional header fields are added to the message.

Sample:
```
IF Subject is *purchase*order*
THEN
Add Header X-Special-Processing: order
```

The Add Header action can be used to add an X-Color field. This field is detected by the WebUser Interface and is used to highlight a message in the Mailbox:

Sample:
```
IF Header Field is X-Spam: *
THEN
Add Header X-Color: red
```

`Tag Subject` *tag*

This action specifies a string to be added to the Subject header field.

When a message is stored, sent, copied, or sent to an external program, the specified subject tags are inserted into the beginning of the Subject header field.

Sample:
```
IF From is ceo@mycompany.dom
THEN
Tag Subject [CEO]
```

If several Tag Subject actions have been used with one message, the latest tag is added first, followed with the other tags, followed with the original message Subject.

**Note:** the following actions are not implicit "Discard" actions, and they do not prevent the original message from being stored in the INBOX. If you want, for example, to redirect a message without keeping a copy in your INBOX, specify the Redirect action followed with the Discard action.

`Store in` *mailboxName*

The message is copied to the specified Mailbox in your Account.

Sample:

```
IF From is developer@partner.com
THEN
Store in DeveloperBox
Discard
```

If the mailbox name starts with the `[MUSTEXIST]` prefix, the prefix is removed, and the Rule processing fails if the Mailbox does not exist. Otherwise, if the Mailbox does not exist, an attempt to create it is made.
If the mailbox name starts with the `[IFEXISTS]` prefix, the prefix is removed, and the action completes immediately if the Mailbox does not exist.

If the mailbox name is specified as `~accountName/mailboxName`, or as `~accountName@domainName/mailboxName` the message is stored in the *mailboxName* Mailbox in the *accountName* Account in the same Domain or in the *accountName@domainName* Account.

When this action is used in a Server-wide or Cluster-wide Rule, the mailbox name must be specified in this form, as there is no default Account for those Rules.

You should have the Insert access right to that Mailbox.

Sample:

```
IF Subject is *Make*$*
THEN
Store in ~postmaster/abuse
Discard
```

If the specified Mailbox cannot be opened or the message cannot be stored in that Mailbox, the Rules processing stops (as if the Stop Processing action was used).

Redirect to *addresses*
> The message is redirected to one or several specified E-mail addresses. If several addresses are specified, they should be separated with the comma (`,`) symbol.
> The specified addresses replace the message To/Cc header fields, unless these specified addresses are prefixed with the `[bcc]` string;
> The "new sender" address is constructed as the E-mail address of the current Account, or to the MAILER-DAEMON address if the action is used in a Server-wide or a Cluster-wide Rule.
> The redirected message Return-path address is set to the "new sender" address, or to the empty address if the Return-path address of the original message was empty.
> The redirected message Sender address is set to the "new sender" address.
> A Return-Path header field (if any) is changed to the X-Original-Return-Path field.
> Return-Receipt-To, Errors-To and DKIM-Signature fields are removed.
> Message-ID, Date, and Sender fields (if any) are renamed into X-Original-Message-ID, X-Original-Date, and X-Original-Sender.
> New Date and Message-ID fields are created.

Forward to *addresses*
> The message is forwarded to the specified addresses. Same as the Redirect operation, but the "new sender" address is not stored as the Sender field. Instead it is used to compose a new From field.
> The old From field is renamed into X-Original-From field.

Mirror to *addresses*
> The message is mirrored (redirected) to the specified addresses (with minimal header changes).

The redirected message Return-Path is preserved.

A Resent-From header field is added. It contains the E-mail address of the current Account (without its Real Name), or the MAILER-DAEMON address if the action is used in a Server-wide or a Cluster-wide Rule.

A Return-Path header field (if any) is changed to the X-Original-Return-Path field.

Return-Receipt-To and the Errors-To fields are removed.

`Reply with` *`message text`*

The specified text is used to compose a reply message. The reply is sent to the address specified in the Reply-To address of the original message. If the Reply-To header is absent, the reply is sent to the From address of the original message.

The header fields `Subject: Re:` *`original message subject`* and `In-Reply-To:` *`original message-ID`* are added to the reply message.

The specified message text can contain macro symbols that are substituted with actual data when a reply message is composed:

`^S` is substituted with the Subject of the original message (in its original form)
`^s` is substituted with the Subject of the original message (in the MIME-decoded form)
`^F` is substituted with the From address of the original message (in its original form)
`^f` is substituted with the From address of the original message (in the MIME-decoded form)
`^T` is substituted with the Date field of the original message
`^I` is substituted with the Message-ID field of the original message
`^R` is substituted with the To field of the original message (in the MIME-decoded form)
`^r` is substituted with the E-mail address of the current Account.

Sample:

```
Reply with
```

If the specified text starts with the plus (+) symbol, the lines following this symbol are added to the message header. The text should specify the Subject field, since the system will not automatically add the `Subject: Re:` *`original subject`* and `In-Reply-To:` *`original message-ID`* fields into the reply message.

The specified header portion can contain additional `To`, `Cc`, and `Bcc` fields and the reply message will be sent to those addresses (the `Bcc` fields will be removed from the message header).

Unless the specified header contains the `From` field, the From field is composed using the From Address from the Account WebUser Settings. If that address is not set the From Address is composed using the full Account Name and the Account Real Name setting.

If the full Account name is not stored as the `From` field, it is stored as the `Sender` field.

The `^S` and other macro symbols can be used in the additional header fields, too.

An empty line should separate the message body from the additional header fields:

```
Reply with
```

If the specified text starts with the `[charsetName]` string, the text is converted to the specified charset (all non-ASCII texts are stored in the UTF-8 charset), otherwise it is converted into the charset used in the incoming message. If the incoming message did not have a charset specified, and the Rule is an Account-Level one, the Preferred Charset specified in the Account WebUser Preferences is used.

If the text starts with the plus symbol, the plus symbol must be specified after the `[charsetName]` string.

Unless the specified header contains the `MIME-Version` and `Content-Type` fields, these fields are added to the composed message.

`Reply to All with` *message text*
> The same as above, but the reply is sent to all addresses listed in the `To` and `Cc` fields of the original message.

`React with` *message text*
> The specified message text should contain a header, an empty line, and the message body. The header should contain any number of `To`, `Cc`, and `Bcc` fields, the Subject field, as well as any number of additional fields. The composed message is sent to the specified addresses.
>
> The specified message header and the message body can contain macro symbols listed above.
>
> The `From`, `Sender`, `MIME-Version`, and `Content-Type` fields are composed in the same way as for the Reply With operation.
>
> Sample:

```
React with
```

The message text can start with the `[charsetName]` string (see above).

Sample:

```
React with
```
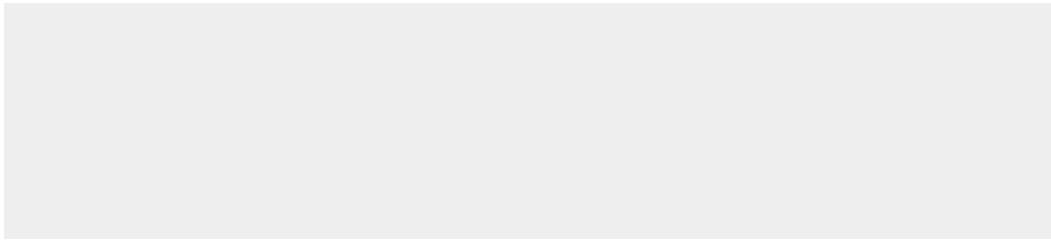
`Store Encrypted in` *`mailbox name`*
> This action works in the same way as the `Store in` action, but a message is converted into [S/MIME encrypted](#) form before being stored.

`Copy Attachments into` *`file directory name`*
> This action copies the message attachments to the specified [File Storage](#) directory.
> Attachments are detected as parts of the topmost `multipart/mixed` or `multipart/related` MIME structure. If a message contains only one non-multipart part, and the Content-Disposition for that part is "attachment", it is treated as an attachment, too.
> If the directory name has the `[replace]` prefix, an existing file with the same name is replaced, otherwise an error is generated if the file already exists.
> If the directory name is empty, then files are stored to the topmost level of the Account File Storage.
> If the directory name ends with the star (`*`) symbol, the symbol is replaced with a unique string, the file extension from the attachment name (if any) is added and the resulting name is used as the File Storage file name for the attachment.

> Sample:

```
Copy Attachments into
```

> Sample:

```
Copy Attachments into
```

`Execute` *`command line`*
> The specified command is executed in a separate OS process (task).
> The message text (the header and the body) is sent to the task as that task *standard input (stdin)*.
> **Note:** the task must read the entire *stdin* data stream, otherwise the Execute command fails.

> A command text can be prefixed with the `[FILE]` tag:

> `[FILE] myprogram parm1`
> When this prefix is used, the task standard input will be empty (closed) or it will contain only the message header fields added by previous Rules.
> The string `-f Queue/`*`fileid`*`.msg` (the -f flag and the Message file name, relative to the *base directory*) will be appended to the end of the command text:
> `-f Queue/12002345.msg`

**Note:** usually access to the *base directory* is not granted to regular users, so the `[FILE]` prefix can be used in the Server-Wide Rules only.

A command text can be prefixed with the `[RETPATH]` tag:

`[RETPATH] myprogram parm1`
When this prefix is specified, the `-p` string followed by the message return-path address is added to the end of the command text:
`-p "address@domain.com"`

A command text can be prefixed with the `[RCPT]` tag:

`[RCPT] myprogram parm1`
When this prefix is specified the `-r` string followed by the list of message recipient addresses is added to the end of the command text:
`-r "address1@domain1.com" "address2@domain2.com"`

A command text can be prefixed with the `[ORCPT]` tag:

`[ORCPT] myprogram parm1`
When this prefix is specified the `-r` string followed by the list of message recipient addresses is added to the end of the command text. If a recipient address was submitted together with its "original recipient" parameter (the ESMTP ORCPT parameter), the original address is used.
`-r "origAddress1@domain1.com" "origAddress2@domain2.com"`

**Note:** the `[RCPT]` and `[ORCPT]` prefixes cannot be used together.

A command text in an Account-Level Rule can be prefixed with the `[ACCNT]` tag:

`[ACCNT] myprogram parm1`
When this prefix is specified the `-u` string followed by the Account name is added to the command text:
`-u "accountName@domainName"`

A command text in a Server-Wide Rule can be prefixed with the `[ROUTE]` tag:

`[ROUTE] myprogram parm1`
When this prefix is specified the string `-R` followed by the Routed recipient addresses is added to the command text:
`-R "LOCAL(accountName@domainName)" "SMTP(example.com)user@example.com"`
See the conditions section for the Route data format information.

A command text can be prefixed with the `[STDERR]` tag (see below).

A command text can have several prefix strings, and they can be specified in any order. If several of `[FILE]`, `[RETPATH]`, and `[RCPT]` prefix strings are specified, the `-f` flag and its parameter are added first, followed with the `-p` flag and its parameter, followed with the `-r` flag and its parameters.

When the task completes, the task *exit code* is checked. If the code is zero, the Rule action is considered as executed successfully, and the next Rule action is executed.

If the task exit code is non-zero, the message is rejected with the error code "`automated processing failed`", and the data from the task *standard error* channel is recorded in the Log along with the task exit code.
If the `[STDERR]` prefix was specified on the command line, the data written to the *standard error* channel (if any) is used to compose the error report text.

The data from the task *standard output*, if any, should not exceed 4Kbytes in size. It is recorded in the Log and discarded.

The CommuniGate Pro Server monitors the task during its execution, and it interrupts the task if it does not

complete within 2 minutes.

When a task is to be executed as a part of Account-Level Rule processing, the OS User Name is composed using the Account [OS User Name](#) setting, and the task is executed in that *OS User Environment*.

> When the CommuniGate Pro runs under control of a Unix system, the task is assigned the specified Unix User *ID*, *group ID*, and the *set of groups*; the task current directory is set to the Unix User *home directory*.

> The Execute action cannot be used in Account-Level Rules if the CommuniGate Pro Server runs under MS Windows, IBM OS/2, AS/400, or BeOS operating systems.

When a task is to be executed as a part of a Server-Wide Rule, it is launched in the CommuniGate Pro Server own environment (with the *base directory* being the current directory).

Sample:

```
Execute
```

ExternalFilter
> This action tells the Server to pass the message to an [External Filter](#) program. This action can be specified only in the Server-Wide Rules. The action parameter specifies the name of the External Filter program to use.

> Sample:

```
ExternalFilter
```

Accept Request *options*
> This action can be used to accept Calendaring Meeting Requests automatically. See the [Calendaring](#) section for more details.

Accept Reply
> This action can be used to accept Calendaring Meeting Replies automatically. See the [Calendaring](#) section for more details.

---

# Macro Substitution

Parameter strings for some actions can include "macro" - symbol combinations that are substituted with actual data before the parameter is used with the Rule action.

The following symbol combinations are available:

^S is substituted with the Subject of the original message (in its original form)
^s is substituted with the Subject of the original message (in the MIME-decoded form, converted to UTF-8)
^F is substituted with the From address of the original message (decoded, converted to UTF-8, including the "Real name" part)
^E is substituted with the From address of the original message (the E-mail address only)
^T is substituted with the RFC822-formatted timestamp taken from the Date field of the original message.
^t is substituted with the RFC822-formatted current-time.
^I is substituted with the Message-ID field of the original message
^R is substituted with the To E-mail addresses of the original message (a comma-separated list)
^r is substituted with the recipient E-mail addresses (a comma-separated list).
^^ is substituted with a single ^ symbol.

---

# Vacation Message

Each Account can have a "simplified" Rule to generate Vacation messages. When enabled, the Rule checks that the message is not an auto-generated one, and that the message author (the 'From' address) has not be placed into the `RepliedAddresses` string list. It then composes and sends an Vacation message and adds the message author address into the `RepliedAddresses` string list, so the Vacation message will be sent to each message author only once. Alternatively, the Notify option may be enabled, so a copy of each incoming message is forwarded to the specified addresses.

This Rule conditions are:

```
Human Generated
Current Date      is greater    specified time (optional)
Current Date      is less       specified time (optional)
From              not in        #RepliedAddresses (optional)
```

The Rule actions are:

```
Reply with           Reply Text
Forward to           address list
Remember 'From' in   RepliedAddresses
```

Only the text of the Reply message can be modified:

| Vacation Message | | | | | |
|---|---|---|---|---|---|
| Starts: | 1 | Aug | 2015 | Ends: | 6 | Aug | 2015 |
| | | | Notify: | | | | |

Vacation Message

If this option is not selected the Vacation Message Rule is disabled. If this option is selected, the Vacation Message Rule is enabled with a low priority (the rule priority is set to 2).

`Starts`

If this option is selected, the Vacation Message Rule starts working at the date specified with this setting.

`Ends`

If this option is selected, the Vacation Message Rule stops working at the date specified with this setting.

`Notify`

If this option is selected and a list of addresses is specified, a copy of incoming message is forwarded to specified addresses. This option is available only when the account has rights to specify the *redirecting* Rule actions.

Even if the Administrator has not allowed the user to specify Automated Rules, the Vacation Message can be enabled by the user herself, and the user can always modify the Vacation Message text.

If "Clear 'Replied Addresses' List" button is clicked, the `RepliedAddresses` string list is removed from the Account dataset. Alternatively, the Enable Vacation Message button can be present. It enables the Vacation Rule and clears the Replied Addresses list at the same time.

# Copy All Mail Simplified Rule

An Account can use a simplified Rule to send copies all incoming E-mail messages to a different address or addresses.

This Rule condition is either empty (the Rule action is applied to all messages) or, optionally, `human generated`, the Rule actions are `Forward to` or `Redirect to` or `Mirror to`, and, optionally, `Discard`.

Only the list of redirection addresses can be modified:

| Copy All Mail To |
|---|
| |

Keep the Original      Do not Copy Automatic Messages      Send on behalf of this Account

`Copy All Mail To`

If this option is not selected the Copy All Mail Rule is disabled. If this option is selected, the Copy All Mail Rule is enabled with the lowest priority (the rule priority is set to 1).

`Keep the Original`

If this option is not selected, the action `Discard` is added to the Rule and all copied messages are NOT stored in the account INBOX.

`Do not Copy Automatic Messages`

If this option is selected, the condition `Human Generated` is added to the Rule and messages from non-human sources (mailing list messages, error messages, redirected and mirrored messages) are not processed with this Rule.

`Send on behalf of this Account`

If this option is selected, the `Forward to` action is used for this Rule. This option must be used if the Account has 'From' Address/Name [restrictions](#) enforced, or if the target host enforces DMARC or other checks of the sender address.

```
Keep the Original sender Address
```
If this option is selected, the `Redirect to` action is used for this Rule.

```
Send unchanged Copy
```
If this option is selected, the `Mirror to` action is used for this Rule.

The account user can set this Rule only if the Account is granted a right to specify the *redirecting* Rule actions. Otherwise only the Administrator can set this Rule for the user account.

# Junk Processing Simplified Rules

An Account can have simplified Rules to handle junk E-mail messages.

| Junk Mail Control | | |
|---|---|---|
| High probability:   Discard | Medium probability:   Store in Junk | Low probability:   Mark as Junk |

These Rules rely on other Rules, usually employing [External Filter](#) programs, to add a special message headers field with the "junk probability" level.

Each Junk Control Rule has a condition checking contents on the `X-Junk-Score` message header field. The other condition checks if the message From address is not included into the AddressBook dataset ("whitelisting").

If the conditions are met, a Junk Control Rule can either discard the E-mail message, store it in the Junk Mailbox (the Junk Mailbox name as specified using the Account Preferences), or mark the stored E-mail Message with the `Junk` flag.

# Logging Rules Activity

The [Enqueuer](#) component records Server-Wide Rules activity in the Log.
Set the Enqueuer Log Level to `Low-Level` or `All Info` to see the Rules checked and the actions executed.

The [Local Delivery](#) module records Domain-Wide and Account-Level Rules activity in the Log.
Set the Local Delivery module Log Level to `Low-Level` or `All Info` to see the Rules checked and the actions executed.

| Overview | Rules | **Filters** | SMTP | Local | RPOP | LIST | PIPE |
|---|---|---|---|---|---|---|---|

# External Filters

- **Starting External Filters**
- **Using External Filters**

This section explains how CommuniGate Pro can employ External Filter programs to scan messages. This feature is used to implement virus protection and content filtering.

The CommuniGate Pro Filters provide a much more solid solution than various stand-alone SMTP-based "mail scanners":

- Stand-alone "scanner" SMTP relays usually implement only the basic SMTP functions. Since all SMTP connections have to be established to those relays, and not to the CommuniGate Pro SMTP module, the CommuniGate Pro SMTP extended functionality becomes unavailable to users and other SMTP servers.
- Stand-alone "scanner" SMTP relays usually provide much weaker performance and reliability than CommuniGate Pro Servers. When the "scanner" relay goes down, the CommuniGate Pro SMTP functionality becomes unavailable, too.
- Stand-alone "scanner" SMTP relays usually cannot scan several messages simultaneously, so when a large message is being scanned, the SMTP traffic to the CommuniGate Pro Server stops.
- Stand-alone "scanner" SMTP relays cannot scan messages not submitted via SMTP. For example, messages composed using the WebUser Interface and directed to a user on the same CommuniGate Pro Server are delivered without any SMTP transfer operations.

External Filters run alongside the CommuniGate Pro Server. They do not deal with message transfer protocols. Instead, the CommuniGate Pro Server passes them a message file right before the message is being enqueued into module queues. As a result, all messages can be scanned, not only the messages sent via a particular mail transfer protocol.

If the CommuniGate Pro ENQUEUER is configured to use several processors (threads), several messages can be scanned simultaneously. As a result, long messages that require several seconds of scanning time do not stop the message flow.

The third-party Plugins distributed by CommuniGate Systems usually require an additional License Key. Several Plugins are currently available.

The Helpers section specifies the information about the External Filters protocol. Read that section if you plan to design a new Plugin.

# Starting External Filters

After you have installed an External Filter program, or built your own one, use the CommuniGate Pro WebAdmin Interface to configure the External Filters. Open the General pages in the Settings realm, and click the Helpers link.

| | | | |
|---|---|---|---|
| Enabled | | | |
| Log Level: | Low Level | Program Path: | |
| Time-out: | 30 sec | Auto-Restart: | 15 sec |
| Disabled | | | |
| Log Level: | Low Level | Program Path: | |
| Time-out: | 60 sec | Auto-Restart: | 60 sec |
| Disabled | | | |
| Log Level: | Problems | Program Path: | |
| Time-out: | Never | Auto-Restart: | Never |

To specify a new External Filter program to run, use the last element in this table. Assign some name to the Filter program and enter into the first field. You will use this name when you specify the `ExternalFilter` Rule actions. Enter the program path and other options, and click the Update button.

To remove an External Filter program, enter an empty string into its Filter name field, and click the Update button.

Each External Filter program has the following options:

`Log`
> Use this setting to specify the type of information the External Filter module should put in the Server Log. Usually you should use the `Problems` Log level (status change and non-fatal errors). But when you experience problems with the External Filter program, you may want to set the Log setting to `Low-Level` or `All Info`: in this case the inter-program protocol-level details will be recorded in the System Log as well.
> The External Filter records in the System Log are marked with the `EXTFILTER` tag.

`Program Path`
> Use this setting to specify the file name path for the External Filter program (with optional parameters). If the External Filter Software has been installed inside the CommuniGate Pro *base directory*, you can use the relative path (`CGPMcAfee\CGPMcAfee.exe`, for example). Otherwise, use the full path (such as `D:\Programs\CGPMcAfee\CGPMcAfee.exe` or `/usr/sbin/myFilter`).
> **Note:** always use the backslash (`\`) path separators if the CommuniGate Pro Server runs on a Microsoft Windows platform.
> **Note:** on Unix platforms, if you want to specify parameters that include spaces or other special symbols, enclose them into the quote (`"`) symbols. On other platforms, use the platform-specific agreements for command line parameters.

Set the first option value to Enabled, and click the Update button to start the External Filter program. If the program cannot be started, an error message appears on the Helpers page.

`Time-out`
> Certain conditions and/or errors in the External program code can make it enter a loop and stop responding to CommuniGate Pro Server requests. If a response for any of the Server requests is not received within the specified period of time, the Server sends a termination signal to the External Program.

`Auto-Restart`
> Certain conditions and/or errors in the External program code can crash that program. Also, the Server itself can send a termination signal to the External program if the program does not respond to requests within the specified period of time (see above).

If the Auto-Restart parameter is not set to Never, the CommuniGate Pro server detects the External Program termination, waits for the specified period of time, and then restarts the External Program automatically. Then it resends all pending requests to the newly started External Program and resumes normal request processing. If the Auto-Restart parameter is set to Never, you need to open the Helpers WebAdmin page and click the Update button to force the Server to restart the External program.

# Using External Filters

An enabled External Filter is not used for scanning mail messages by default. If you have specified an External Filter program with the *filterName* name, you can scan all messages with that program by creating a Server-Wide [Rule](#). Specify no condition for that Rule (so the Rule will apply to all messages the Server processes), and specify one Rule action - ExternalFilter *filterName*.

Messages are scanned only when the option next to the Filter name is set to Enabled. You may want set this option to Disabled to let messages bypass this External Filter program. If this option is set to Disabled, the ExternalFilter *filterName* Rule operation is a null operation (it does nothing).

If you want to scan only some messages, add condition(s) to this Rule. The following sample Rule check the size of a message, and uses the McAfee External Filter program to scan only those messages that are larger than the specified limit:

| Data | Operation | Parameter |
|---|---|---|
| Message Size | greater than | |
| --- | is | |
| **Action** | **Parameter** | |
| ExternalFilter | | |
| --- | | |

External Filters are contacted from the Server [ENQUEUER](#) threads. Since it can take several seconds to process a large message, increase the number of ENQUEUER processors (threads) using the Queue page in the WebAdmin Interface Settings realm.

Alternatively, you can disable the Enqueue Asynchronously option (on the same page), and make each submitting thread scan the messages during the submit process.

| Overview | Rules | Filters | SMTP | Local | RPOP | LIST | PIPE |

# SMTP Module

- **Simple Mail Transfer Protocol (SMTP) and DNS**
- **Configuring the SMTP module**
- **Sending Messages via the Internet**
  - **Sending via a Forwarding Mail Server**
  - **Sending Directly to the Recipients**
  - **Multi-channel Delivery**
  - **Sending via Dial-up Links**
  - **Retrying Sending Attempts**
  - **Secure (encrypted) Message Relaying**
- **Receiving Messages**
  - **Sender Authentication**
  - **Limits and Protection**
  - **Waking up the Backup Server**
  - **On-demand Mail Relaying (ATRN)**
  - **LMTP Support**
- **Serving Dial-up Client Hosts**
  - **Remote Queue Starting (ETRN)**
  - **On-demand Mail Relaying (ATRN/TURN)**
  - **Waking up via E-mail**
  - **Holding Mail in Queue**
- **Message Relaying**
  - **Relaying via Dedicated IP Addresses**
  - **Processing the Submit Port**
- **Processing Mail from Blacklisted Addresses**
- **Routing**
- **Sending to Non-Standard Ports**
- **Monitoring SMTP Activity**

The CommuniGate Pro SMTP module implements E-mail message transfer using the SMTP and ESMTP Internet protocols via TCP/IP networks.

The Simple Mail Transfer Protocol allows computers to transfer messages using network connections. A computer that has a message to send connects to the recipient's computer and establishes a network link. Then it sends one or several messages and closes the network connection.

Mailer applications use the SMTP protocol to submit messages to the mail servers, and mail servers then forward the submitted messages to the recipients. All mailers have a setting called SMTP Host Address that specifies the network address of the mail server computer. Mailer applications open TCP connections to that address when they have a message to submit.

The CommuniGate Pro SMTP module supports special "secure ports" and the STARTTLS SMTP extension, and it can receive and send mail via secure (encrypted) connections.

The CommuniGate Pro SMTP module supports the AUTH extension and it allows remote users to authenticate themselves before submitting messages.

The CommuniGate Pro SMTP module also implements a protocol variation called LMTP (Local Mail Transfer Protocol).

# Simple Mail Transfer Protocol (SMTP) and DNS

Mail servers use the global Domain Name System to find the network address of the recipient computer or the recipient mail server. Each domain (part of the E-mail address after the @ symbol) should have a special so-called MX-record in the Domain Name System. That record specifies the name of the computer that actually receives mail for that domain. For example, MX records can specify that mail for the domain `company.com` should be sent to the computer `mail.company.com`, and mail to the domain `enduser.com` should be sent to the computer `provider.com`.

There can be several MX-records for one domain (with different priority values). If one (high-priority or primary) computer cannot receive mail, mail is sent to lower-priority computers (called Back-up Mail Servers). Back-up mailer servers then try to deliver the message to the primary server.

When the name of the recipient computer is retrieved from the DNS, the sending mail server consults the DNS again. Now it uses the DNS to convert the receiving mail server name into its network address. The so-called DNS A-records contain the pairs that link a computer name to its global Internet network (IP) address.

When the network address of the recipient mail server is received from the DNS, the sending mail server opens an SMTP connection to that server and transfers the message(s). When all messages to that domain are transferred, the connection is closed.

When a message contains several addresses within the same domain, the SMTP module can transfer only one copy of the message to the mail server serving that domain, and that server delivers messages to all recipients in that domain. But if there are too many addresses, the SMTP module can break them in several portions and send several copies, each containing only a portion of the address set.

If there are several messages to one domain, the SMTP module can open several connections to the mail server serving that domain and send those messages simultaneously.

If you want to receive messages from the Internet with your own mail server, you should register your domain name, and ask your provider to register that name with the Domain Name System. The DNS records should point to the computer running your mail server.

# Configuring the SMTP module

Use the WebAdmin Interface to configure the SMTP module. Open the Mail pages in the Settings realm, then open the SMTP pages.

| Processing | |
|---|---|
| Log Level:   All Info | Channels:   30 |

Use the Log setting to specify what kind of information the SMTP module should put in the Server Log. Usually you should use the `Major` (message transfer reports) or `Problems` (message transfer and non-fatal errors) levels. But when you experience problems with the SMTP module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.

The SMTP module records in the System Log are marked with the `SMTPI` tag for incoming connections, with the `SMTP` tag for outgoing connections, and with `SMTPW` tag for connections used to wake up the back-up server.

# Sending Messages via the Internet

If you want to send messages over the Internet, your server should have a TCP/IP link to the Internet. When a message should be transferred to some remote host, the SMTP module connects to that host via the TCP/IP network, and it transfers the message using the SMTP protocol.

`Channels`
> This setting (see above) limits the number of outgoing SMTP connections the module is allowed to open simultaneously.

| Processing | | |
|---|---|---|
| Send Directly to Recipients | Default IP Address: | OS default |
| Forward to | | |
| Channels/Host:  4 | Add Channels after: | 60 sec |
| | or when Queue size is: | 50 |
| Recipients/Message:  unlimited | Hide 'Received' fields | |
| | Always try 'EHLO' | |

`Default IP Address`
> When CommuniGate Pro receives a message, it can assign one of its local network addresses to that message, and if the SMTP module needs to send such a message, it will send it via the assigned local address. See the [Domains](#) section for more details. All other messages are sent out via the Default SMTP IP Address. Set this setting to `OS Default` to let the server OS pick the proper local network address itself.

`Recipients/Message`
> This setting specifies how many recipients can be sent with one message. If a message has too many recipients, the specified number of recipients is sent, then the message body is sent, and then the module repeats the sending procedure with the same message - for the remaining recipient addresses.

`Hide Received Fields`
> If this option is selected, the SMTP module modifies all `Received` header fields in the messages it sends to non-client network addresses. Only the time stamp values are left in those fields, while all other field values are replaced with some dummy data.

`Always use EHLO`
> If this option is selected, the SMTP module always sends the `EHLO` command to remote servers, trying to establish the extended SMTP (ESMTP) protocol.
> If this option is not selected, the SMTP module checks the remote server greeting line. The SMTP module sends the `EHLO` command only if this line contains the `ESMTP` word.

When sending messages over the Internet, the SMTP module can forward them to some other mail server, or it can deliver messages directly to the recipients, using the DNS MX-records to find the recipient hosts on the Internet.

When the SMTP module sends a message from a Domain with an assigned IP address:

- the connection to the remote server can be made from the Domain IP address, see [Domain Settings](#) section
- the module sends the HELO/EHLO command using that Domain name. If this name does not resolve to the Network IP Address of this CommuniGate Pro Server, some receiving systems may reject SMTP connections

coming from your Server.

In this situation you can update the [Domain Settings](#) to specify an alternative name to use with the HELO/EHLO command.

When the SMTP module sends a message from a Domain without any assigned IP address:

- the connection to the remote server is made from the [Default IP address](#)
- the module sends the HELO/EHLO command using the Main Domain name. If this name does not resolve to the Network IP Address of this CommuniGate Pro Server, some receiving systems may reject SMTP connections coming from your Server.

  In this situation you can update the Default [Domain Settings](#) to specify an alternative name to use with the HELO/EHLO command.

# Sending via a Forwarding Mail Server

`Forward to`

When this option is selected, the SMTP module connects to the specified *forwarding mail server* (also called a *smart host*) and sends all queued messages to that server. Since the forwarding mail server is usually "close" to your server, messages leave your system quickly. But this method can cause additional delays in message delivery, since messages are queued on the forwarding server and those queues can be processed slowly. This method is recommended when your server is connected to the Internet using a slow link, or when you use a dial-up link and you want messages to leave your server as soon as possible to keep the connection time short.

Select the Forward to option and specify the name or the IP address of the forwarding server. The SMTP module will forward all outgoing messages to that mail server for delivery.

**Note:** the name of the forwarding mail server should be the name of the real computer (as specified in an A-type DNS record), not a mail domain (MX-type) name. While your provider domain name can be `provider.com`, the name of the provider mail server can be something like `mail.provider.com`. Consult with your provider to get the exact name of the forwarding mail server you can use.

**Note:** you can specify the IP address of the forwarding server instead of its name. You can also specify several IP addresses, separated with the comma (,) symbol. If an SMTP connection to the first specified address cannot be established, the SMTP module will try the next specified address.

**Note:** when configuring a [Cluster Backend](#) Server, you can specify the asterisk (`*`) symbol in the Forward To setting. In this case, all Cluster Frontends (specified on the Cluster Settings page) will be used as the forwarding mail servers.

**Note:** when a recipient domain name is specified as an IP address (as in `user@[12.34.56.78]`), the SMTP module delivers messages directly to the host with the IP address 12.34.56.78, even if the `Forward to` option is selected. You may use this feature for message exchange between several mail servers on a LAN that does not have its own Domain Name Server.

`AUTH Name`

The forwarding mail server should be configured to enable relaying from your server to any other server on the Internet. Some forwarding mail servers may require your server to use the AUTH command with a valid name and password parameters before transferring messages that need to be relayed. In this case you should enter the AUTH login name and password in the AUTH fields.

This type of configuration is used when your server has a "dynamic IP address", and receives mail from the same forwarding server using the [ATRN method](#). Usually the username and password used for mail forwarding are the same as the username and password used for ATRN receiving.

# Sending Messages Directly to Recipients

```
Directly to Recipients
```
When this option is selected, the SMTP module uses the DNS (Domain Name System) to convert message recipient addresses into the names and addresses of the receiving hosts. A receiving host can be the recipient host itself, or a relay host. The information about the proper relay host is stored in so-called MX records on Domain Name Servers. For each destination host several records can exist, each record having a priority value. If the SMTP module fails to connect to the relay host with the highest priority, other MX records are used and other relay hosts are tried. If no relay host is available, the message remains in the SMTP queue, and more attempts to deliver it (and all other messages to the same host) are made later.

This method allows the system to deliver a message either directly to the recipient computer or to a relay host that is "very close" to the recipient computer. Recipients can read your messages almost immediately, and your messaging system does not rely on any "forwarding mail server" performance.

## Multi-channel Delivery

When the Server queue contains several messages to be directed to the same domain, the SMTP module opens a connection to that domain mail server and sends messages one by one. If the established connection is slow and there is a large message in the Queue, other messages would wait too long before being delivered. You may want to allow the SMTP module to open additional connections to the same mail server and send other messages in parallel.

```
Channels/Host
```
Use this setting to limit the number of TCP connection the SMTP module is allowed to establish with one domain mail server.

```
Add Channel after
```
When the SMTP module cannot send a single message within this time period, and there are other messages in the same Queue, the module opens a new parallel connection.

```
when Queue size is
```
If the SMTP module is sending a message, and there number of unsent messages in the same Queue is equal or larger than the value of this setting, the module opens a new parallel connection.

## Sending via Dial-up Links

The SMTP module sending activity can be limited using the TCP Activity Schedule. Outgoing messages wait in the SMTP queue till the TCP Activity Schedule allows the Server to initiate outgoing network connections.

When outgoing activity is allowed, the SMTP module tries to send all submitted messages accumulated in its queue.

## Retrying Sending Attempts

If an attempt to deliver a message fails, the SMTP module can delay delivery of that message, or it can delay delivery of the entire host queue (if a connection with the host could not be established or an established connection was broken, etc). The Retrying panel allows you to control how the SMTP module makes attempts to deliver messages:

| Retrying | | | |
|---|---|---|---|
| Retry Every: | 30 min | for the first: | 3 hour(s) |
| then Every: | 3 hour(s) | for: | 3 day(s) |
| Messages with empty 'Return-Path': | | for: | 3 hour(s) |
| Delay Messages: | 20 min | Delay Recipients: | 20 min |
| | | Send Warnings after: | 3 hour(s) |

**Retry Every**

Use this setting to tell the SMTP module when it should retry to send a message if a connection fails. If you use a forwarding mail server, this option specifies when the module should retry to connect to that server if the previous connection failed for any reason. If you use the Directly to Recipients method, and a connection to some remote host (domain) fails, all messages directed to that domain will be suspended for the specified time. Usually SMTP systems suspend messages for 30 minutes.

**for the first .... then Retry**

Use these settings to tell the SMTP module when and how it should change the retry interval. Usually, you would tell the SMTP module to increase the retry interval to 1-2 hours after a message has spent more than 2 hours in the queue.

**for**

Use this setting to limit the number of attempts to deliver a message. If a message cannot be delivered within the specified period of time, the message is rejected and an error report saying that the host is unavailable is sent back to the message sender.

**Messages with empty 'Return Paths': for**

This setting specified the alternative Keep Trying value for messages with empty Return-Paths. These messages are used to report failed or successful delivery, and they can be discarded from the Queue after fewer attempts.

**Delay Messages**

When a receiving host returns a "temporary failure" (4xx) response for a message sending command (MAIL FROM, DATA, the "final dot"), this setting specifies when an attempt to send this message should be repeated.

**Delay Recipient**

When a receiving host returns a "temporary failure" (4xx) response for a message recipient command (RCPT TO), this setting specifies when an attempt to send the message to this recipient should be repeated.

**Send Warnings After**

Use these settings to tell the SMTP module when warning messages should be sent back to the message senders, notifying them about delivery delays.

# Secure (encrypted) Message Relaying

You can configure your CommuniGate Pro Server SMTP module to use secure (encrypted) connections when sending messages to certain remote sites. This feature is especially useful if your company has several offices and E-mail traffic between the offices is sent via the public Internet.

You should simply list the domain names that should receive mail from your server via secure connections:

| Send Encrypted (SSL/TLS) |
| --- |
| to Domains: (high security) |
| wherever possible (low security) |

The specified names can contain a wildcard - the asterisk (*) symbol.

When the CommuniGate Pro SMTP module connects to a relay of one of the listed domains, it checks if that relay supports the STARTTLS protocol extension command. Then the SMTP module uses this command to initiate a secure connection with that relay.

The CommuniGate Pro SMTP module checks the validity of the remote relay Certificate using the specified set of the Trusted Certificates.

The remote relay Certificate *subject* must contain the `cn` (*Common Name*) field that matches either the domain name of the remote site, or the name of this relay. This can often cause a problem, since the domain `company.dom` may have the MX record `relay1.company.dom`, but the computer with the `relay1.company.dom` address has the "main" DNS name `smtp.company.dom` and its Certificate is issued to that name (its Certificate *subject* contains `smtp.company.dom` in the `cn` field).

To solve this problem, you should explicitly route all traffic to the `company.dom` domain via the `smtp.company.dom` relay, using the following [Router](#) record:

```
NoRelay:company.dom = company.dom@smtp.company.dom._via
```

See the [Routing](#) section for more details about SMTP routing.

**Note:** this feature ensures that messages between your server and a remote relay are transferred securely. To provide complete end-to-end security, you should verify that:

- users submit messages to servers either using a private network, or using TLS/SSL connections over the public Internet (secure SMTP or secure WebMail);
- all mail servers and relays exchange messages either using a private network, or using TLS/SSL connections over the public Internet (secure SMTP);
- users read messages either using a private network, or using TLS/SSL connections over the public Internet (secure POP, IMAP, WebMail).

If the domain is listed in the Send Secure To Domains list, and the receiving server does not support the STARTTLS command, or the remote server certificate cannot be validated, or the remote server certificate Subject does not match the domain or domain relay name, all messages to that domain are **rejected**, ensuring that no message is sent via a potentially insecure link.

If your server sends all outgoing mail via a forwarding server, you can enter the asterisk (`*`) symbol into the [Send Encrypted](#) field to encrypt all communications with the forwarding server.
The CommuniGate Pro SMTP module does not check the *Subject* of the forwarding server certificate.

`wherever possible`
> Select this option if you want the SMTP module to try to use SSL/TLS connections with all remote SMTP servers that support this feature. If the remote domain is **not** listed in the Send Secure To Domains list, but the remote server supports the STARTTLS command, the SMTP module tries to establish a secure (SSL/TLS) connection with that server.
> The module does not check the remote server Certificate validity or the Certificate Subject in this case. If the STARTTLS command or secure connection negotiations fail, the server defaults back to plain-text communication and sends messages via an unencrypted channel.
> Some servers advertise STARTTLS support, but fail to accept SSL/TLS connections. When this option is selected, it is impossible to send E-mail to those servers. To solve this problem, inform the broken server administrator, and enter the server domain into the Send Secure To Domains list, prefixed with the exclamation point (`!`) symbol. The SMTP module will not try to use SSL/TLS connections with that server/domain.

If an outgoing connection is made to the port 465 (see [Sending to Non-Standard Ports](#) section), then the SMTP module initiates the secure (SSL/TLS) protocol immediately after establishing a TCP/IP connection.

---

# Receiving Messages

The SMTP protocol is used to receive messages from the Internet and from the client mailer applications. If you want to receive messages from the Internet, you need a TCP/IP link to the Internet, and your server domain name and the IP address should be included into the DNS records.

Click the Receiving link on any SMTP Settings page to open the SMTP Receiving settings page.

| Processing | | |
|---|---|---|
| Log Level:   All Info | **Listener** | Channels:   25 |

**Channels**

When you specify a non-zero value for this setting, the SMTP module creates a so-called "listener". The module starts to accept all SMTP connections that other mail servers establish in order to send mail to your Server. This setting is used to limit the number of simultaneous connections the SMTP module can accept. If there are too many incoming connections open, the module will reject new connections, and sending mail systems will retry later.

**Listener**

This link allows you to tune the SMTP Listener. You can specify which TCP ports to use for SMTP incoming connections (by default, the port 25 is used), which local IP addresses to use for incoming connections (all available addresses are used by default), and which remote addresses should be granted access to your CommuniGate Pro SMTP Server (by default, all addresses can connect to the SMTP port).
**Note:** to allow Microsoft® Outlook Express 4.x users to submit messages using secure connections, you should configure the SMTP listener to accept connections on the TCP port 465, and enable the SSL/TLS option for that port.
**Note:** Netscape® Messenger and modern versions of Microsoft Outlook and Outlook Express products do not need any special port for secure communications, since these products use the STARTTLS command to initiate secure communications after establishing a regular, clear text SMTP connection to the standard port number 25.

| Processing | | | |
|---|---|---|---|
| Advertise: | AUTH to:   everybody | | 8BITMIME to:   non-clients |
| | NO-SOLICITING: | | |
| Verify: | Return-Path for:   non-clients | | HELO for:   non-clients |
| | Check SPF records:   Disabled | Reverse Connect:   Disabled | |
| Require: | STARTTLS for:   nobody | | |

**Advertise AUTH capability**

If a server reports (in its initial EHLO prompt) that it supports SMTP Authentication, some mailer clients force users to authenticate themselves before sending messages. If you select the Non-Clients value, and a connection is accepted from an address included into the Client IP Addresses list, the SMTP module will not report that it supports the AUTH command. If the Nobody value is selected, the SMTP module never reports that it supports SMTP AUTH. This option does not disable the SMTP Authentication feature itself.

**Advertise 8BITMIME**

If your Server does not report the 8BITMIME capability, some mailers and servers will MIME-encode all non-ASCII messages that they send to your Server . This server-side encoding can cause troubles for many old mail clients. To avoid these troubles, your Server should report the 8BITMIME capability.

**Note:** The CommuniGate Pro SMTP module itself never converts non-ASCII messages into the MIME form, and (according to RFC1652) it should not advertise the 8BITMIME capability. But the modern Internet is completely 8-bit transparent and clean, so it is safe to enable the Advertise 8BITMIME option, preventing other servers from doing unnecessary 8bit-to-MIME message body conversion.

**Advertise NO-SOLICITING**

Use this setting to specify the *Solicitation class keywords* your Server is not willing to accept. See the RFC3865 for more details.

Verify HELO
This option specifies if the parameters of HELO/EHLO commands should be verified.
You can tell the module to verify these addresses only for the connections from network addresses not included into the Client IP Addresses list. This is useful if:

- many of your clients use mailers that send bogus names in the `HELO/EHLO` commands;
- your Internet connection is a dial-up one, and you do not want any outgoing (DNS) traffic to be generated when receiving mail from your own client computers (Client Hosts).

Verify Return-Path
Return-Path E-mail addresses (sent using the `MAIL FROM` protocol commands) are processed with the [Router](#). If the resulting address is routed to a remote host (a non-local domain name), this option specifies if the that domain name should be verified.
See the [Protection](#) section for more details.

Check SPF records
This option tells the SMTP module to verify non-local Return-Path domain names using the SPF DNS records.

- If this option is set to Disabled, the SPF records are not used.
- If this option is set to "Add Header", the SPF records are checked, and the `Received-SPF` header field with the SPF record processing result is added to all incoming messages.
- If this option is set to Enabled, the SPF records are checked.
  If these records say that messages with the specified Return-Path domain cannot be sent from the sender network (IP) address (the SPF records specify "hard failure" for that address), the Return-Path is rejected.
  In other cases, the `Received-SPF` header field is added to the message.
- If this option is set to "with DMARC", the SPF records are checked, Return-Path and message body are accepted;
  Then if the sender is not authenticated then domain in 'From:' address from the message header is checked; if the domain does not exist (no A nor MX DNS records) the message body is rejected.
  Then DMARC record for the domain is checked, if the record is not found then the record for the organizational domain is checked. If the DMARC policy is "reject" and SPF check result is not "pass" and the message has no DKIM signature which is valid and aligned to the domain, the message body is rejected.
  In other cases, the `Received-SPF` header field is added to the message.

**Note:** the SPF/DMARC records are not checked for SMTP connections from [Client](#) or [White Hole](#) network addresses.
**Note:** If the Verify Return-Path option is set to Nobody, this option is not used.

Reverse Connect
This option tells the SMTP module to verify non-local Return-Path addresses by connecting to the mail servers hosting those addresses and verifying that those servers accept the specified addresses.
If this option is set to Add Header, the addresses that cannot be verified are not rejected. Instead, the `X-Reverse-Check` header field containing the verification failure code is added to the message.
**Note:** Reverse Connect processing is not used for incoming SMTP connections from [Client](#) or [White Hole](#) network addresses.
**Note:** If the Verify Return-Path option is set to Nobody, this option is not used.

Require STARTTLS
This option is applied to messages with Return-Path addresses not belonging to any Domain created on this Server.
If the SMTP connection is established from the specified address (Client, non-Client, etc.) and the connection is not secured using SSL/TLS encryption, the Return-Path and the message itself is rejected.
For messages with Return-Path addresses belonging to some Domain created on this Server, this functionality is controlled using the [Domain Settings](#).

# Sender Authentication

If a message sender (the message Return-Path specified with the MAIL FROM protocol command) is a local Object - an Account, or a Group, or a Mailing List in a local Domain, that Domain is opened, and its SMTP Force AUTH option is checked.

If this option is enabled, the message will be rejected if the client mailer has not sent the SMTP AUTH command first. The option value specifies for which sending mailer IP addresses this feature should be used.
**Note:** this option checks for the "fixed" Client IP Addresses only - it does not pay attention to the "temp-client" addresses added with the Process as a Client Address feature.

**Note:** use this option carefully. Some users may use different mail relays to submit their messages with their CommuniGate Pro Account names as the message Return-Paths.
If this option is enabled and those messages are directed to your Server, they will be rejected, because mail relay servers are not able to authenticate the senders on your Server.

**Note:** most mailers will send the AUTH command only when the server advertises its SMTP AUTH capability. Make sure that your server does advertise it (see above).

# Limits and Protection

| Limits | | | | |
|---|---|---|---|---|
| Message Limits: | Size: | 10 Mbytes | Recipients: | 25 |
| Non-Client Sender: | Recipients: | 10 | in: | 30 seconds |
| | Delay Prompt for: | 30 seconds | | |
| | Disconnect after: | 20 errors and | Deny Access for: | 15 minutes |

Message Size Limit
: This setting tells the module to reject all incoming messages that are larger than the specified limit.

Message Recipients Limit
: This setting limits the number of recipients for each incoming message. Specifying a lower value makes your server less attractive for spammers.

Non-Client Sender: Limits
: Use this option to specify a time period and the limit on the number of message recipients. For each non-client IP address the SMTP module counts all messages coming from that IP address, and all recipients specified for those messages. If the total number of all recipients specified during the set time period exceeds the set limit, new recipients are rejected with a temporary code, forcing the sending server to retry sending messages to those recipients later.
If a sender has completed a successful SMTP AUTH operation that allows the sender to relay via the CommuniGate Pro server, the recipients sent via that connection are not counted.
This setting can be used to protect your Server from spammers that send one message or a batch of messages to a large number of users of your system. On the other hand, if a large number of your users is subscribed to some mailing list, this option can cause delays for messages coming from that list.

Non-Client Sender: Delay Prompt
: Use this option to specify the delay time between the moment when an SMTP connection is accepted from a non-client address and the moment when the Server responds with an initial prompt. This delay can be used to force spam-sending programs to disconnect from your Server. Normal servers should still be able to transfer mail, as the RFC2821 standard requires them to wait 5 minutes for the initial prompt, and most servers do wait at least 3 minutes.
When this setting has a non-zero value, the Server checks if any data line has been received from the sender before

the initial SMTP prompt is sent. If the sender has sent some data without waiting for an initial Server prompt, an error message is returned to the sender and the connection is closed.
The Prompt Delay is introduced only for connection made to the port 25.
**Note:** if you start to use long Prompt Delays, expect to see your Server using much more SMTP Input channels, as each SMTP transaction becomes longer.

`Non-Client Sender: Disconnect`
Use this option to specify how many protocol errors the SMTP module should detect before it drops the connection with the sender. This feature protects your Server from spammers that try various E-mail addresses (dictionary attack), causing "unknown account" errors.
The SMTP module places the network addresses of disconnected senders into the Temporarily Blocked Addresses list. The SMTP module rejects new connections from the Temporarily Blocked Addresses.
**Note:** the module does not block a "failed sender" network address if that address is a Client IP Address (specified explicitly or via DNS) or if that address is a White Hole Address.

When an incoming recipient address (`RCPT TO`) command addresses a local Account, the Account Domain settings can instruct the SMTP Module to check the current status of that Account.
If E-mail to the Account can not be delivered immediately (Account is over quota, etc.), the recipient address is rejected with a "temporary failure" (4xx) response code.

## Waking up the Backup Server

If your Server has a dial-up link, its domain name should have at least one additional DNS MX record, specifying a "back-up" mail server (usually, your ISP mail server). When your Server is off-line, all messages directed to your domain(s) are sent to that back-up mail server.

The back-up mail server tries to deliver collected messages to your server. Usually, the retry period is 30 minutes, so your system should stay on-line for at least that period of time in order to receive messages from the back-up server.

To avoid this delay, the SMTP module can be configured to send the Remote Queue Starting ("ETRN") command to the back-up server. When the back-up server receives that command, it immediately starts to send the collected messages to your Server.

| Retrieving from a Backup Server | | | |
|---|---|---|---|
| Send Wakeups | Every: | minute | to: |
| Use ATRN | AUTH Name: | | Password: |

`Send Wakeups`
Use these settings to specify the address of the Back-up Server, and to specify how often the Remote Queue Starting command should be sent.

**Note:** the name of the back-up server should be the name of the real computer (as specified in an A-type DNS record), not a mail domain name. While your provider domain name can be `provider.com`, the name of the provider mail server can be something like `mail.provider.com`. Consult with your provider to get the exact name of your back-up server, or just examine the DNS MX records for your domain: your back-up server is specified with the MX record that has the priority next to your own Server MX Record priority.

The SMTP module wake-up activity is limited with the TCP Activity Schedule.

## On-demand Mail Relaying (ATRN)

The `ETRN` command can be used to release your domain queue on a remote backup server only if your server has a static

IP address.

If your server has a dynamic IP address, the ETRN method does not work, since the backup server does not know the IP address your server is using, and the backup server is not able to open a connection to your server.

If your server uses a dynamic IP address, it should use the On-demand Mail Relaying method to retrieve mail from the backup server.

When On-demand Mail Relaying method is used, your server connects to the backup server, authenticates itself, and then it issues the ATRN command. Then the servers exchange their roles and the backup server starts to send your server your domain mail via the same channel. This eliminates a need for the backup server to open a connection to your server.

Since the backup server does not open a connection itself, it has to verify that the server that sends the ATRN command and wants to retrieve your domain mail is really your server. Your server should provide some name and password that should be accepted by the remote server and that should allow your server to issue the ATRN command.

Consult the remote server administrator to learn the name and the password your server should send before sending the ATRN command.

Use ATRN
 select this option to use the ATRN (On-demand Mail Relaying) method instead of the ETRN method.

AUTH Name and Password
 this pair of strings is sent to the remote server using the AUTH command. The access rights granted to this login name on the remote server should allow your server to use the ATRN command.

The CommuniGate Pro SMTP module uses the AUTH CRAM-MD5 authentication method to send passwords in an encrypted form. If the remote server does not support the CRAM-MD5 method, the clear-text AUTH LOGIN method is used.

If your backup server does not support On-demand Mail Relaying, you should use the Unified Domain-Wide Account method implemented with the RPOP module.

The RFC2645 suggests to use the special TCP port number 366 to provide the ATRN services. If your backup mail server provides the ATRN services on that port (or on any port other that the standard SMTP port 25), you should specify the port number in the Send Wakeups To setting field. Use the colon symbol to separate the server name and the port number:
mail.provider.dom:366

You can use secure communications with the backup server if you include the backup server name into the Send Encrypted list.

When the backup server name is specified, the SMTP Settings page displays the Wake Up Now button. Click that button to initiate a wakeup session immediately.

## LMTP Support

The SMTP module implements the LMTP protocol. It supports this protocol on all its ports, and it is not required to configure additional ports just to support LMTP.

The SMTP module switches to the LMTP mode when it receives the LHLO LMTP command.

# Serving Dial-up Client Hosts

The CommuniGate Pro Server can be used as a back-up mail server for dial-up systems. Dial-up systems receiving mail via SMTP expect their back-up servers to receive and keep all their messages when these systems are off-line. When a dial-up system connects to the Internet again, it connects to its back-up mail server and either issues the special Remote Queue Starting command (ETRN, RFC1985), or sends a dummy E-mail message to a special address on the back-up server.

# Remote Queue Starting (ETRN)

When your server receives the ETRN command, it tries to send out all messages collected for the host specified as the ETRN command parameter. This method allows a dial-up system to get its messages immediately, instead of waiting for your server to make the next attempt to deliver the collected messages.

The SMTP module supports the ETRN command, so CommuniGate Pro can be used as a back-up mail server. No special setting is required, since this feature is always enabled.

The SMTP module uses the [Router](#) to process the ETRN parameter (domain name). It adds the `wakeup` fictitious user name to that domain to get a regular E-mail address `wakeup@etrn-parameter` and runs it through the Router. If the address is routed to an SMTP host, the SMTP module releases (wakes up) that host queue.

If you have routed the domain `client.com` to `mail.client.com` in your Router Table, all mail to the client.com domain will be kept in the mail.client.com queue. Since the ETRN command parameter is processed with the Router, too, the `ETRN client.com` command will correctly release the mail.client.com queue.

In a [Dynamic Cluster](#) environment, the ETRN command received by any cluster member releases domain queues on all cluster members.

# On-Demand Mail Relaying (ATRN/TURN)

When the SMTP module receives the `ATRN` command, it checks that the connected party has authenticated itself. Then the module releases the specified domain queue and sends all its messages directly via this (already established) connection. No special settings is required to enable the ATRN feature of the CommuniGate Pro SMTP module. There are some notes about the ATRN implementation:

- Only one ATRN command parameter is allowed.
- The name of the domain queue to be released should match the name of the authenticated user. If you want to allow a dial-up client host to release the `domain.dom` queue, you should create the `domain.dom` Account in the CommuniGate Pro Main Domain, and the client host should authenticate itself using the domain.dom as the login name and the domain.dom Account password as the password.
- If the ATRN command does not have a parameter, the name of the authenticated user is used as the name of the queue to be released.
- The domain name used in the ATRN command must be included into the `Hold Mail for Domains` list.

The `ATRN` command parameter (if any) is processed in the same way the [ETRN](#) command parameter is processed.

The RFC2645 suggests to use the special TCP port number 366 to provide the ATRN services. CommuniGate Pro SMTP module provides the ATRN services on all ports its [Listener](#) is using. To comply with the RFC2645 standard, you may want to add the port 366 to the SMTP Listener settings.

For compatibility with legacy Microsoft Exchange servers, the `TURN` SMTP command is supported, too. It is processed in the same way as the ATRN command without a parameter, and it requires authentication, too. The name of the queue to release is the same as the name of the authenticated user.

# Waking up via E-mail

The SMTP module supports an alternative wakeup method: a dial-up system can send any message to *domainName-*

`wakeup@`*`serverDomain`* to release the domainName message queue. The serverDomain name should be the Main Domain name of the CommuniGate Pro Server.

In a [Dynamic Cluster](#) environment, the Wakeup E-mail received by any cluster member releases domain queues on all cluster members.

## Holding Mail in Queue

You can ask the SMTP module to hold mail for certain hosts in its queue, and not to try to deliver that mail until the receiving server issues the ETRN command or sends a wake-up E-mail. This can be useful if the receiving server is on a symmetric dial-on-demand line and its provider brings the link up automatically when there is any traffic for that receiving server.

---

# Message Relaying

The situation when the SMTP module receives a message from a remote system and then sends that message to some other host is called *relaying*.

To avoid Server abuse, some relay restrictions can be specified.

| Relay | | | | |
|---|---|---|---|---|
| Relay to any IP Address: | If Received from: | clients | IP Address | |
| Relay to Client IP Addresses: | If Sent to: | simple | E-mail Address | |
| Relay to Hosts We Backup: | | Enabled | | |

| ETRN/ATRN Processing |
|---|
| Accept Wakeups from:    clients |
| Hold Mail for Domains: |
| |

Relay to any IP Address
> See the [Protection](#) chapter for the information about this setting. If the Nobody option is selected, relaying is still possible for authenticated users.

Relay to Client IP Addresses
> See the [Protection](#) chapter for the information about this setting.

Relay to Hosts We Backup
> This option allows the SMTP module to relay messages to any domain, if the MX records for that domain includes this CommuniGate Pro Server name (its Main Domain name) as a back-up mail relay.

Accept Wakeups
> This option tells the SMTP module to accept ETRN commands and wake-up E-mail messages either from anybody, or only from the hosts included into the Client IP Addresses list, or from no host at all.
> Since the ATRN command, unlike ETRN, requires authentication, the ATRN command is always accepted from any address.

Hold Mail for Domains
> When the SMTP module builds a queue for one of the domains (hosts) listed in this field, it immediately places that queue "on hold", waiting for the ETRN/ATRN command or any other external action that releases that queue. This method should be used for the sites that receive mail via your server, and that want to receive it only when they issue the ETRN/ATRN command.

The specified names can contain a wildcard - the asterisk (`*`) symbol.

# Relaying via Dedicated IP Addresses

You may want to send messages for some of your CommuniGate Pro Domains via Local IP Addresses assigned to those Domains. See the [Domain Settings](#) section for more details.

If a message is to be delivered to the *hostName* host via a particular *12.34.56.78* Local IP Address, the message is not placed into the `hostName` SMTP queue. Instead, the Server places it into the `@12.34.56.78|`*hostName* SMTP queue.

This technique allows the Server to process messages from different Domains independently. If the IP Address of one of your Domains is blacklisted by remote hosts (because that Domain users have abused the mail system), messages to the same remote hosts from other Domains will not be delayed or rejected.

If the `hostname` name is included into the [Hold Mail for Domains](#) list, the Server never adds the `@12.34.56.78|` prefix, so all messages to that host are always enqueued into a single queue, and that queue can be used with the ATRN command.

The ETRN command releases not only the `hostname` queue, but all *hostName* queues with preferred IP prefixes (`@12.34.56.78|`*hostName* queues).

## Processing the Submit Port

You may want to enable SMTP receiving on the port 587. Sessions established to that port are processed differently:

- Connections are not rejected when the [Reserved for Clients](#) limit is reached.
- The AUTH capabilities are always advertised.
- Messages (the MAIL FROM commands) are accepted only after successful authentication.

---

# Processing Mail from Blacklisted Addresses

When a [Blacklisted](#) host connects to the SMTP module, the module does not reject a connection. Instead, it receives the `MAIL FROM` SMTP command, and starts to process the recipient (`RCPT TO`) addresses sent from the blacklisted host. The module adds the domain `blacklisted` to each recipient address received from a blacklisted host, i.e. the received address `user@domain` is converted into `user%domain@blacklisted`. Then the address is processed with the Router as usual. If the Router Table does not contain special rules for the `blacklisted` domain, the address is rejected with a special error code.

The default Router Table contains the following line:
`<blacklist-admin*@blacklisted> = postmaster`
All messages from blacklisted hosts sent to the `blacklist-admin` address in any domain, are routed to the postmaster, so these messages are accepted. This "white hole" feature allows the blacklisted host users to contact the postmaster on your server if they want to discuss the blacklisting issue. If you remove this line from the Router Table, no address will be accepted from blacklisted hosts.

When rejecting addresses sent from blacklisted hosts, the SMTP module verifies if the `blacklist-admin@blacklisted` address can be routed with the Router. If the Router Table contains such records (a default one or a different one), the error code sent back to the blacklisted host explains that mail to `blacklist-admin@serverdomain name` is accepted even from that blacklisted site.

If you want to provide a "white hole" feature, but you do not want the information about the white-hole address to be included into the error code, simply use a different name for the "white hole" address.

Example:

```
        <abuse*@blacklisted> = postmaster
```

The following table contains samples of SMTP sessions established from a blacklisted host. The host commands are marked with `c:`, the SMTP module responses are marked with `s:`.

| Router Table | |
|---|---|
| **SMTP protocol** | `C: MAIL FROM: user@host`<br>`S: 250 user@host sender accepted`<br>`C: RCPT TO: somebody@somehost`<br>`S: 591 Your host [10.1.1.1] is blacklisted. No mail will be accepted`<br>`C: RCPT TO: abuse@somehost`<br>`S: 591 Your host [10.1.1.1] is blacklisted. No mail will be accepted`<br>`C: RCPT TO: blacklist-admin@somehost`<br>`S: 591 Your host [10.1.1.1] is blacklisted. No mail will be accepted`<br>`....` |

| Router Table | `<abuse*@blacklisted> = postmaster` |
|---|---|
| **SMTP protocol** | `C: MAIL FROM: user@host`<br>`S: 250 user@host sender accepted`<br>`C: RCPT TO: somebody@somehost`<br>`S: 591 Your host [10.1.1.1] is blacklisted. No mail will be accepted`<br>`C: RCPT TO: abuse@somehost`<br>`S: 250 abuse%somehost@blacklisted will leave Internet`<br>`C: RCPT TO: blacklist-admin@somehost`<br>`S: 591 Your host is blacklisted. No mail will be accepted`<br>`....` |

| Router Table | `<blacklist-admin*@blacklisted> = postmaster` |
|---|---|
| **SMTP protocol** | `C: MAIL FROM: user@host`<br>`S: 250 user@host sender accepted`<br>`C: RCPT TO: somebody@somehost`<br>`S: 591 Your host [10.1.1.1] is blacklisted. Send your questions to blacklist-admin@mycompany.com.`<br>`C: RCPT TO: abuse@somehost`<br>`S: 591 Your host [10.1.1.1] is blacklisted. Send your questions to blacklist-admin@mycompany.com.`<br>`C: RCPT TO: blacklist-admin@somehost`<br>`S: 250 blacklist-admin%somehost@blacklisted will leave Internet`<br>`....` |

# Routing

The SMTP module immediately (on the first [Router](#) call) accepts messages addressed to `domain name-wakeup` local addresses. When these messages are enqueued into the SMTP module queue, they are processed as [wake-up requests](#) for the domain name domain message queue.

The SMTP module also immediately accepts all addresses with IP-address domains, i.e. with domain names like `[xx.yy.zz.tt]`. Please note that the [Router](#) adds brackets to the IP-address domain names that do not have them, and the Router changes the IP addresses of local domains to those domain names. The Router performs these operations before calling the modules.

The SMTP module immediately accepts addresses that have domain parts ending with the `._smtp` suffix. This suffix is processed in the same way as the [._via](#) suffix.

The SMTP module immediately accepts addresses that have domain parts ending with the `._smtpq` suffix. This suffix is processed in the same way as the [._relay](#) suffix.

On the final call, the SMTP module accepts mail to any domain if that domain name contains at least one dot (`.`) symbol. If the Forward option is selected, all these addresses are rerouted to the specified Forwarding Server domain.

Before accepting an address, the SMTP module checks if the address does not contain any `@` symbol, but contains one or several `%` symbols. In this case, the rightmost `%` symbol is changed to the `@` symbol.

# Sending to Non-Standard Ports

Some mail servers can be configured to receive incoming SMTP mail on a non-standard port. The CommuniGate Pro SMTP module can send messages to those servers, if the domain part of an E-mail address contains the port number or is routed to an address that includes the port number.

Use the `._relay` or `._via` suffix with the port number: `domain.port._relay`. The SMTP module will not use the *domain* MX records in this case, it will try to resolve the *domain* name directly into an IP address. Sample Router record:
`secret.communigate.com = mail.communigate.com.26._relay`

# Monitoring SMTP Activity

The Monitors realm of the WebAdmin Interface allows Server Administrators to monitor the SMTP module activity. The SMTP Monitor section contains three pages - the Delivery (sending) page, the Waiting page, and the Receiving page.

The Delivery page displays the active outgoing SMTP connections:

| ID | Target | Messages | Status | Running | Message | Size | Return-Path |
|----|--------|----------|--------|---------|---------|------|-------------|
| 158996 | domain1.dom | 1 | opening connection | 8sec | 23458798 | | |
| 158997 | domain2.dom | 2 | opening connection | 2sec | 58798234 | | |

ID
: This field contains the numeric ID of the outgoing SMTP session. In the CommuniGate Pro Log, this session records are marked with the SMTP-*nnnnn* flag, where *nnnnn* is the session ID.

Target
: This field contains the target Queue name (which normally matches the destination hostname).

Messages
: This field contains the number of messages in the destination Queue.

Status
: This field contains the operation in progress.

Running
: If there is an SMTP operation in progress, this field contains the time since operation started.

Message, Size, Return-Path
: If there is a message being sent to the destination host, these fields contain its ID, size and Return-Path, respectively.

The Receiving page displays the incoming SMTP connections:

| ID | Source | Account | Status | Running | Size | Return-Path | Recipient |
|----|--------|---------|--------|---------|------|-------------|-----------|
| 151 | [192.168.0.104]:58312 | user1@domain1.dom | reading recipients | | 28798 | user1@domain1.dom | user2@domain1.dom |
| 152 | [10.0.0.104]:47531 (TLS) | | checking return-path | 2sec | | xfs@company.com | |

ID
: This field contains the numeric ID of the incoming SMTP session. In the CommuniGate Pro Log, this session records are marked with the SMTPI-*nnnnn* flag, where *nnnnn* is the session ID.

Source
: This field contains the IP address the sender has connected from.
: The (TLS) mark indicates that the SSL/TLS encryption is being used.

`Account`
     This field contains the name of the client Account, if the sender had authenticated.

`Status`
     This field contains the operation in progress.

`Running`
     If there is an SMTP operation in progress, this field contains the time since operation started.

`Size, Return-Path, Recipient`
     If there is a message being received, these fields contain its size, Return-Path or a recipient address, respectively.


If the sender had connected to port 586 (SMTP Submit), the connection row is displayed with a green background.

SMTP activity can be monitored using the CommuniGate Pro Statistic Elements.

| Overview | Rules | Filters | SMTP | Local | RPOP | LIST | PIPE |

## Navigation Sidebar

Introduction
Installation
SysAdmin
Network
Objects
Storage
E-mail
Real-Time
Access
Services
Directory
Clusters
Applications
Miscellaneous
Licensing

WebMail
Pronto!
PBX

# Local Delivery Module

- **Configuring the Local Delivery Module**
- **Incoming Message Flow Control**
- **Routing**
- **Routing to Unknown Accounts**
- **Unified Domain-Wide Accounts**
- **Automated Mail Processing**
- **Storing Mail in Account Mailboxes**
- **Direct Mailbox Addressing**
- **Routing Settings**
- **Sending Mail to All Accounts**

The Local Delivery module processes messages routed to Accounts on your Server. It uses the Mailbox Manager to store messages in Account Mailboxes.

The Local module applies Account Automated Mail Processing Rules to all messages directed to that Account. Rules can instruct the module to store a message in a different Mailbox, to redirect a message to different address(es), etc.

After a message is stored in an Account Mailbox, it can be retrieved using any of Access modules.

The Local Delivery module can support Direct Mailbox addressing and Account Detail addressing.

The module can limit the number of messages each Account can receive during the specified period of time. This feature allows the Server to minimize damage caused by mail loops.

# Configuring the Local Delivery Module

Use the WebAdmin Interface to configure the Local Delivery module. Open the Mail pages in the Settings realm, the open the LOCAL pages.

| Processing | | |
|---|---|---|
| Log Level: All Info | | Processors: 3 |
| If Mail Service is disabled: | Retry Every 20 min | for: 2 day(s) |
| If Account is full: | Retry Every 60 min | for: 24 hour(s) |
| Suspend: | Account Queue | Message |
| | Send Warnings after: 10 min | |

Log
Use the Log setting to specify what kind of information the Local Delivery module should put in the Server Log. Usually you should use the `Major` (message transfer reports) or `Problems` (message transfer and non-fatal errors) levels. But when you experience problems with the Local Delivery module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.
The Local Delivery module records in the System Log are marked with the `LOCAL` tag.

Processors
When you specify a non-zero value for this setting, the Local Delivery module starts to process queued messages directed to local Accounts. The module can use several simultaneous processors (threads) to deliver messages to several Accounts at the same time. If you have more than 1000 Accounts, or if you have many Accounts with time-consuming automated Rules, you should allow the module to use more than 1 processor for message delivery.

If the Account is full
When you specify a non-zero value for this Retry setting, the Local Delivery module checks the Account mail storage before trying to deliver a message to that Account. If the Account mail storage size is limited, and the specified percent of that limit is already used, or it would be used when the new message is added, the Local Delivery module delays all messages directed to that Account. The module retries to deliver mail to that Account periodically. It resumes message delivery when some messages are deleted from the Account Mailboxes.
This parameter specifies how long incoming messages should be kept in the module queue before they are rejected with the `account is full` error message. If the Retry value is smaller than the Every value, messages are not kept in the Queue when an Account is full: they are rejected immediately.

If the Mail Service is disabled
When you specify a non-zero value for this setting, messages sent to an Account with disabled Mail Service option are not rejected immediately, but they are left in the module queue for the specified period of time. The module tries to deliver the delayed messages periodically. When the administrator re-enables the Account/Domain Mail Service, the queued messages are delivered to the Account.

Suspend
If the incoming message size is larger than the he Account mail storage limit, the message is rejected.
If the Account mail storage limit does not allow an incoming message to be stored, but the message size itself does not exceed that limit, the message is delayed.
This setting specifies if the entire message queue for that Account should be suspended (till some messages are removed from that Account or its quota is increased), or that only this message should be suspended individually (so smaller messages from the same Account queue can be delivered).

Send Warnings after
If a message is delayed in the module queue (because the addressed Account is full or the Mail Service is disabled for that Account), the module can generate a warning message and send it back to the message sender. Use this setting to specify when the warning message should be generated.

# Incoming Message Flow Control

While CommuniGate Pro employs many built-in techniques to prevent mail loops, in some situations (usually involving other servers) mailing loops still can occur. To minimize the damage caused by those loops, the Local Delivery Module counts all messages received by each Account. If this number exceeds the specified limit, the incoming messages queue for that Account is suspended.
**Note:** The module counts the number of messages to be delivered to the Account, not the number of messages stored: even if an incoming message is not stored in the Account INBOX because an Account Rule has discarded it, the message is still counted.

The Account [Mail Transfer Settings](#) specify the incoming flow control limits for each Account.

---

# Routing

When the [Router](#) passes an address to the Local Delivery module, the module checks the domain name: if the domain name ends with the string `.local`, the Local Delivery module accepts the address, removes the `.local` suffix from the domain name, and stores the message in the Main Domain Account with that name. This feature is used to create [Unified Domain-Wide Accounts](#).

Example:

a message sent to the address

`abcdef@nnnnn.local`

will be accepted with the Local Delivery module, and the message will be stored in the `nnnnn` Account.

Sometimes, a Unified Domain-Wide Account should be created in a Secondary Domain, rather than in the Server Main Domain. Use the `.domain` suffix to direct mail to an Account in a secondary Domain. The last component of the address "local part" will be used to specify the name of the Secondary Domain Account:

Example:

a message directed to the address

`abcdef%xyz@nnnnn.domain`

will be accepted with the Local Delivery module, and the message will be stored in the `xyz` Account in the `nnnnn` Domain.

When the Router calls the Local Delivery module for "first attempt", the module does not process any other addresses.

When the Router calls the Local Delivery module for "final delivery" attempt, it accepts all addresses with an empty domain name part or with the domain part equal to the name of a [Secondary Domain](#), and it routes the messages to the Account specified with the "local part" of the address.

Examples:

a message sent to the address

`abcdef`

will be accepted with the Local Delivery module, and this message will be stored in the `abcdef` Account in the Main Domain.

if subdomain.com is one of the Secondary Domains, a message sent to the address

`xyz@subdomain.com`

will be accepted with the Local Delivery module, and this message will be stored in the `xyz` Account in the subdomain.com Secondary Domain.

To provide the *domain-only* routing feature used within the [HTTP module](#), the Local Delivery module accepts all addresses with the `LoginPage` local part, and an empty domain part or a domain part equal to some Secondary Domain name or its Domain Alias name.

---

# Routing to Unknown Accounts

When the Local Delivery module decides that an E-mail address is a local address, it checks that the Account with the specified name exists. Each Domain (the Main one and each [Secondary](#) Domain) has a setting that instructs the Local Delivery module on what to do if a specified Account does not exist.

If the selected option is "Rejected", all messages sent to unknown Accounts are rejected, and the error message "unknown account" is returned to the sender.

If the selected option is "Discard", all messages sent to unknown Accounts are rerouted to the NULL address, and the Server discards them without generating any error messages.

If you select the "Reroute to" option, all messages sent to unknown Accounts will be rerouted to the specified address. That address can be a name of a registered local Account, or it can be an E-mail address of an account on another server: the unknown Account address is substituted with the specified address, and the Router restarts the address processing procedure.

The specified "rerouting" address may contain the asterisk (`*`) symbol. In this case the name of the unknown local account is used to substitute the asterisk symbol.

Example:
> the Reroute address is:
> `bad-*@monitoring.department.com`
> a message is sent to
> `james@mycompany.com`
> where mycompany.com is the domain name of your Server, and there is no Account james on your Server.
> The message is rerouted to:
> `bad-james@monitoring.department.com`

---

# Unified Domain-Wide Accounts

The Router can route an entire domain (domain name) to a certain local Account, if the `.local` domain suffix is used (see above).

This method is useful if:

- the domain belongs to a dial-up client system that does not have a static IP address and thus cannot receive its mail via SMTP;
- the domain has only few POP3 users and you do not want to create a full-featured CommuniGate Pro Domain to serve them.

Example:
> The Router line:
> ```
> client1.com = client.local
> ```
> All messages sent to the client1.com domain are directed to the `client1` local Account in the Main domain.

Unified domain-wide Accounts are useful if the client systems retrieves messages from your Server using the CommuniGate Pro RPOP or similar software that distributes retrieved messages locally. Alternatively, the client system can use a regular single-user mailer and then distribute retrieved messages manually.

While the information in the local part of the client1.com addresses is not used for routing, it is not discarded. When the Local Delivery module stores the message in the client1 Account, it stores the local parts of the addresses in the `X-Real-To:` message header field (or other field specified in the Local Delivery module settings).

Example:
> The Router line:
> ```
> client1.com = client1.local
> ```
> A message sent to:
> `abcdef@client1.com, xyz@client1.com`

is stored in the client1 Account, and a header field:

`X-Real-To: abcdef, xyz`

is added to the stored message.

**Note:** the

```
<*@client1.com>= client1
```

Router alias record also stores all messages sent to the client1.com domain in the client1 Account, but if such a record is used, the information about the local part (Account name) would be lost, and no X-Real-To: head fields would be generated. The client software that retrieves messages from this Unified Account would have to rely on the To: and Cc: message header fields. Those fields do not always contain the correct information, and they never reflect any change in the local part of the address you could have done with some additional routing records.

The [POP module](#) allows individual users to retrieve mail from a Unified Account, by hiding out all messages that do not contain the specified username in the X-Real-To header field.

You usually create Unified Domain-Wide Accounts in the Main Domain. Use the `.domain` suffix to create an UDWA in a Secondary Domain. Messages routed to *xxxx%accountname@domainname*`.domain` will be stored in the *accountname* Account in the *domainname* Domain, with the *xxx* address being added to the message headers as the X-Real-To field.
For example, a Domain Administrator for the company.com Domain may use the setting:

   Mail To Unknown Addresses is Redirected to:`*%Unknowns@company.com.domain`

and messages sent to unknown Domain Accounts will be stored in the Account `Unknowns`, with all those unknown addresses stored in the message `X-Real-To` header fields.

---

# Automated Mail Processing

After an address is accepted with the Local Delivery module, the message is queued to the Module queue. Each Module process takes messages from the queue, and opens the addressed Account.

If the Account has some Automated [Rules](#) specified, these Rules are applied: for each rule its conditions are checked, and if they are met, the specified Rule actions are performed. As a result of those actions, the message can be copied to some Mailbox, a copy of the message can be redirected to some other addresses, an automatic reply can be generated, etc.

You can use a more detailed Log Level for the Local Delivery module to see which Rules are applied to messages, why some conditions are not met, and what actions have been taken when all Rule conditions have been met.

---

# Storing Mail in Account Mailboxes

After Account [Rules](#) (if any) have been applied, and these Rules have not specified that the message should be discarded, the message is stored in the Account INBOX.

The Local Delivery module checks the current size of the Account Mailboxes and rejects a message if the Account storage quota would be exceeded.

---

# Direct Mailbox Addressing

The Local Delivery Module can deliver messages directly to non-INBOX Mailboxes. If the local part of the address is specified as *box*`#`*name*, then the message will be stored in the *box* Mailbox in the *name* Account.

The Account-Level rules are NOT applied if such an address is used.

You can use Direct Mailbox Addressing in the [Router](#) Table:

```
; store messages to sales@maindomain
; in the sales Mailbox in the Account public@maindomain
<sales> = sales#public
;
; store messages to support@client.com
; in the requests Mailbox in the Account staff in the hq.client.com Domain
<support@client.com> = "requests#staff"@hq.client.com
```

**Note:** remember that Mailbox names are case-sensitive.

**Note:** the Direct Mailbox Addressing feature can be used via the [POP module](#), too. With the sample Router records listed above, when a user logs in using the name `sales`, the client POP mailer is connected to the Mailbox `sales` in the `public` Account (if the user has provided the correct password for the `public` Account).

---

# Routing Settings

| Routing |
|---|
| Envelope Recipients field: |
|                      Always Add this field |
| Direct Mailbox (`mailbox#account`):   Enabled |
| Account Detail (`account+detail`):   Disabled |

`Envelope Recipients field`
> This setting specifies the name of the message header field the Local module generates when it stores messages in [Unified Accounts](#).

`Always Add this field`
> If this option is selected, the module adds the message header field with the envelope address(es) to all messages stored in local Accounts.

`Direct Mailbox Addressing`
> This setting specifies if [Direct Mailbox addressing](#) is enabled.

`Account Detail`
> This setting controls Account *detail addressing*. Account detail address is an Account name followed with the plus (+) symbol and some string.
> You can set this setting to:

> `Disabled`
>> The Local Delivery module will not process the plus symbols in Account names.

> `Enabled`
>> The Local Delivery module will check for the plus symbol in Account names and delete the first plus symbol and all following symbols from the address, then it will re-route the address. Users can use Account detail addresses (`john+jokelists`) to subscribe to mailing lists. Messages sent to Account detail addresses will be routed to the user Accounts, and Account-level [Rules](#) (the Recipient condition) can be used to process those message automatically - for example, to store them in some `jokes` Mailbox dedicated to these list messages.

> `Direct Mailbox`
>> The Local Delivery module will check for the plus symbol in Account names and process the string after the plus symbol as the [Direct Mailbox](#) address. The `john+jokelist` address will be processed as the

`jokelist#john` address and the message will be routed directly to the `jokelist` Mailbox of the `john` Account, bypassing Account-Level Rules (if the Direct Mailbox Addressing option is enabled).

# Sending Mail to All Accounts

The [Domain Settings](#) can be used to enable the virtual object `All`. Messages sent to the `all@`*`domainname`* address are stored in INBOXes of all Domain Accounts that have the `Accept Mail To All` option enabled.

**Note:** the individual Account Rules are not applied to messages sent to the `all` address.

The `alldomains@`*`maindomain`* address can be used to send messages to all Accounts in all Domains.

| Overview | Rules | Filters | SMTP | Local | **RPOP** | LIST | PIPE |

# RPOP Module

- **Post Office Protocol (POP3) and Mail Retrieving**
- **Configuring the RPOP Module**
- **Configuring Account RPOP Records**
- **Processing Unified Domain-Wide Accounts**
  - **Mail Distribution without Special Header Fields**
- **RPOP Record Format**
- **Appendix A. Configuring `sendmail` for Unified Domain-Wide Accounts**

The CommuniGate Pro RPOP implements E-mail message retrieval using the POP3 Internet protocol (STD0053) via TCP/IP networks. While the POP Module allows the CommuniGate Pro users to retrieve mail from their CommuniGate Pro Server Mailboxes, the RPOP Module retrieves messages from other (remote) hosts and delivers them to user Mailboxes or to other destinations.

The RPOP Module can retrieve messages for each CommuniGate Pro Account from several remote mailboxes. The RPOP Module can retrieve mail for your entire Domain using "Unified Domain-wide accounts" and distribute retrieved messages to their recipients.

The RPOP Module supports non-standard MSN POP3 servers: if the remote host (server) domain name ends with `.msn.com`, the Module uses the non-standard AUTH MSN method to log into that server.

# Post Office Protocol (POP3) and Mail Retrieving

The RPOP Module can be used when the CommuniGate Pro Server has a dial-up connection with dynamically assigned IP address, and thus the Server cannot receive mail via SMTP. The RPOP Module polls the specified remote host (ISP) accounts, retrieves messages and stores them in the Server mailboxes.

Users with several mail accounts on several systems can instruct the RPOP Module to poll those accounts, so all their mail is collected in their CommuniGate Pro Account.

The RPOP Module supports Unified Domain-Wide accounts. A Domain-wide account is an account on the ISP or any other host that collects all messages sent to your Domain. The RPOP Module retrieves all messages from such an account and distributes them based on the addressing information in the message header fields. The RPOP Module can poll several Unified Domain-Wide accounts.

The RPOP Module activity can be limited using the TCP Activity Schedule. The Module does not poll any remote account till the TCP Activity Schedule allows the Server to initiate outgoing network connections.

# Configuring the RPOP Module

Use the WebAdmin Interface to configure the RPOP Module. Open the Mail pages in the Settings realm, then open the RPOP pages.

| Processing | | |
|---|---|---|
| Log Level: Problems | | Processors: 10 |
| Delay Failed Hosts for: 3 min | Use APOP | |
| Delay Failed Accounts for: 40 min | Allow Self-Poll | |
| Source IP Address: OS default | Use Domain IP Addresses | |

**Log**

Use the Log setting to specify what kind of information the RPOP Module should put in the Server Log. Usually you should use the `Major` (message transfer reports) or `Problems` (message transfer and non-fatal errors) levels. But when you experience problems with the RPOP Module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.

The RPOP Module records in the System Log are marked with the `RPOP` tag.

**Processes**

When you specify a non-zero value for this setting, the RPOP Module starts to connect to the remote hosts and retrieve mail from accounts on those hosts. The setting is used to limit the number of simultaneous connections the RPOP Module can initiate.

**Use APOP**

The RPOP can use the secure APOP authentication method when connecting to hosts that support this feature. If for any reason you want the RPOP Module to always use the "clear text" passwords, disable the Use APOP option.

**Source IP Address**

This option selects the default *source network address* for outgoing POP3 connections. You can allow the server OS to select the proper address or your can explicitly select one of the server IP addresses as the default source network address.

**Use Domain IP Addresses**

This option selects *source network addresses* for outgoing POP3 connections. If this option is selected, the RPOP Module will use the first Assigned IP Address for the Domain the RPOP record belongs to, and if the Domain Assigned IP Addresses can be used for outgoing connections.
If this option is not selected, or if the Domain does not have any Assigned IP Address, the RPOP Module uses the default *source network address*.

**Delay Failed Hosts**

When the RPOP Module fails to connect to an external host, it marks the host as "failed" and stops polling all accounts on that host. The option specifies when the RPOP Module should try to poll the failed host again.

**Delay Failed Account**

When the RPOP Module fails to open a mailbox (wrong password, remote mailbox is locked, etc.), or if the connection fails when the Module retrieves messages from a remote account, the Module delays polling of this Account for the specified period of time.

**Allow Self-Poll**

Very often CommuniGate Pro users misunderstand the concept of remote account polling and specify their own CommuniGate Pro accounts as the "remote" accounts to be polled. This creates message loops and wastes Server resources. If this option is not selected, the RPOP Module checks the network address of the remote POP server it has to connect to. If that address is one of the CommuniGate Pro Server own network addresses, the "remote" account is not polled.

Click the Update button to modify the RPOP Module settings.

# Configuring Account RPOP Records

The CommuniGate Pro RPOP Module can poll POP accounts on remote hosts on behalf of the CommuniGate Pro Account users. For each CommuniGate Pro Account several external POP accounts (RPOP records) can be specified. RPOP records can be specified by Server and Domain Administrators, using the WebAdmin Interface.
Open the Account Settings pages and open the RPOP page in the Mail section:

| Name | Poll Every | Account | at Host | Password | Leave | APOP | TLS | Mailbox | Last |
|------|-----------|---------|---------|----------|-------|------|-----|---------|------|
| firstISP | 30 minutes | | | | | | | | 12:34:56 |
| | Never | | | | | | | | |

If an Account has the CanModifyRPOP setting enabled, the Account user can modify the Account RPOP records via using the WebUser Interface, or a XIMSS client.

Name

 This is the RPOP record name: any text specified when the record was created.

Poll Every

 This option specifies how often the RPOP Module should poll the remote account.
 Set this option to Never to remove this RPOP record.
 If you set this option to Disabled, the RPOP record is not removed, but the remote account is not polled.

Account

 This option specifies the mail account name at the remote host.

at Host

 This option specifies the exact name of the POP server that should be polled. Please note that this could be the name of a specific computer (as specified in DNS A-records), not just a generic domain name of the provider system. For example, if the provider has the domain name provider.com, its POP server is usually named mail.provider.com or pop.provider.com. Consult with your provider.

 **Note:** Standard POP servers accept incoming connections on the TCP port 110. If you need to poll an account on a remote POP server that uses a non-standard port, specify the port number after the host name, using the colon (:) symbol as the separator:
 `pop.provider.com:111`

Password

 The password to use to log into the remote account.

Leave

 If this option is selected, the RPOP Module does not delete messages from the remote mailbox. Instead, it remembers the UID (Unique IDentifier) of the retrieved messages, and the next time the RPOP Module polls this remote account, it does not retrieve messages that have the same UIDs.
 If you want to use this option, verify that the remote POP server supports the `UIDL` command.

 **Note:** messages UIDs are stored in the Account [File Storage](), in the `private/rpopids/name` text files, where *name* is the RPOP record name.

APOP

 If this option is selected AND the UseAPOP RPOP Module option is enabled AND the target host advertises APOP capability in its initial prompt, the RPOP Module uses the secure APOP method for authentication on that remote host.

TLS

 If this option is selected, the RPOP Module tries to establish a secure (SSL/TLS) connection with the remote host.

 **Note:** Standard POP servers accept incoming secure connections on the TCP port 995. If you need to poll an account on a remote secure POP server that uses a non-standard port, specify the port number after the host name, using the colon (:) symbol as the separator:
 `pop.provider.com:9786`

Mailbox

This option can specify some Account Mailbox name. If some name is specified, then the retrieved messages are immediately stored in this Mailbox, without any additional processing.

If this option is not specified, the retrieved messages are sent to the Account via the CommuniGate Pro Queue, so all Server-Wide and Account-Level Rules (including External Filters) are applied to these messages.
These messages are flagged as 'do not report failures', so if delivery to the Account was unsuccessful, no error report is sent to the original message sender.

`Last`

If the last attempt to retrieve mail from the remote account was successful, this field tells when (in the server local time) this attempt took place.
If the last attempt was not successful, the field contains the error code.

To remove an RPOP record, set its polling period value to Never.

To create a new RPOP record, enter its name in the last table row and select some valid polling period value.

Click the Update button to modify the RPOP record set.

---

# Processing Unified Domain-Wide Accounts

A mail account on an external host can collect messages directed to all Accounts (users) of your Domain. The RPOP Module can be instructed to retrieve mail from such an account and distribute it to the local users.

When a message is sent via the Internet, the information about the sender and the message recipients is sent in the so-called mail envelope. If mail is sent via SMTP, the envelope is sent as a sequence of protocol commands.

The information in the envelope is usually the same as the information in the message header fields, but it is not always true. The most important exceptions are:

- the message header fields do not contain the addresses of the Bcc recipients
- the header fields of a mailing-list message do not contain the mailing list subscriber addresses.

When a message is stored in a mailbox, the envelope information about the sender is added to the message headers as the Return-Path header field. Usually, the envelope information about the recipients is not added to the message headers.

When the RPOP Module retrieves a message from a Unified Domain-Wide Account, it has to recompose the message envelope and deliver the message to its final recipient. If the message contains the Return-Path header field, the address in that field is placed in the new envelope as the sender's address, and the header field is removed from the message (it will be recreated when the message is delivered to its final destination).

If a Unified Domain-Wide Account is created with the mail system that can copy the recipient addresses from the envelope into some message header field, then the delivery via RPOP is as reliable as SMTP delivery.
Enter the name of that header field into the Unified Account RPOP record settings, and the RPOP Module will look for that field in all messages retrieved from that account. The addresses from that field will be placed into the new envelope and the messages will be directed to those addresses. The header field itself is removed from the message. All accepted addresses get the 'report on failure' flags, so if message delivery fails, the original message sender (the address in the message Return-Path field) will receive an error report.

Unified Domain-Wide Accounts can be provided with a CommuniGate Pro Server running on the provider side. For messages stored in those accounts, the envelope recipients are added to the message headers as the `X-Real-To` fields. To learn how to provide Unified Domain-Wide Accounts with CommuniGate Pro, check the Local Delivery Module section.

A legacy `sendmail` system can be configured to add `X-Real-To` header fields, too. See the Appendix A below.

RPOP records for Unified Domain-Wide Accounts should be created for the `postmaster` Account in the Main Domain.

The WebAdmin RPOP page for this Account contains the Special field:

| Name | Poll Every | Account | at Host | Password | Leave | APOP | TLS | Special | Last |
|------|-----------|---------|---------|----------|-------|------|-----|---------|------|
| firstISP | 30 minutes | | | | | | | | 12:34:56 |
| | Never | | | | | | | | |

Special
> The name of the messages header (RFC822) field that the provider host inserts into the messages stored in the Unified Domain-Wide Account.

## Mail Distribution without Special Header Fields

Many ISPs still use various legacy mail systems that cannot store envelope recipients in message headers. If you have to host your Unified Domain-Wide Account on such a system, enter the star (*) symbol into the Special field.

The RPOP Module will search for all `To:`, `Cc:`, and `Bcc:` header fields in retrieved messages. It will use the addresses from those header fields only if that address is routed to any existing local CommuniGate Pro Account.

If an address is routed to the SMTP or some other Module, or an address cannot be routed at all (unknown user name error, etc.), the RPOP Module does not send any error messages to the sender. The Module simply ignores that address.

All accepted addresses get the 'do not report failures' flags, so if the message delivery fails for any reason, no error report is sent to the original message sender.

If none of the message `To:`, `Cc:`, or `Bcc:` addresses has been accepted, the RPOP Module sends that message to the `postmaster` Account in the Main Domain.

As explained above, the method based on `To:`/`Cc:` header field parsing can cause problems when the actual envelope addresses are not the same as the header field addresses. Besides, some systems do not process the Unified Accounts correctly, so if a message is sent to three users in your domain, those systems may store three copies of the message in the Unified Domain-Wide Account Mailbox. Since each message header contains the addresses of all three users, the RPOP Module will deliver three copies of the message to each user.

The problems with Bcc, mailing lists, and duplicated message can be very annoying, so we strongly recommend you to ensure that the provider's mail system adds the envelope information to the messages stored in your Unified Domain-Wide Account, and you can use the Special Header Field feature.

## RPOP Record Format

The RPOP records are stored in the Account database as a dictionary. An RPOP record name is used as the dictionary key, and the correspoding value is a dictionary, containing the following elements:

domain
> the domain name or the IP address of the remote mail system.

authName
> the mail account name in the remote mail system.

password
> the mail account password in the remote mail system.

mailbox
> (optional) the name of the mailbox to store retrieved messages in.

special
> (optional) the name of the "envelope address" messages header field (see above).

leave
> (optional) if the element is present and it has the `YES` value, mail messages will not be deleted from the remote mail system after they have been retrieved.

TLS
(optional) if the element is present and it has the YES value, connections to the remote mail system must be established securely, using the SSL/TLS protocol.

APOP
(optional) if the element is present and it has the YES value, the APOP login method should be used with the remote mail system.

# Appendix A. Configuring `sendmail` for Unified Domain-Wide Accounts

The following file can be used to force the freeware `sendmail` program to store the envelope information in message headers.

```
# This file should be placed into the directory cf/feature from
# the sendmail.8.X.XX.cf.tar.Z archive.
# To add special headers, the macros `FEATURE(xrealto)' should be
# added to the main configuration file in the directory cf/cf,
# and the flag T should be added to the mailer description.
#
# This file adds special headers with the `X-Real-To' keyword.
# The special headers will be added to all messages routed to the
# mailer marked with the `T' flag in the sendmail configuration.
divert(0)
VERSIONID(`@(#)xrealto.m4 0.1 1/4/96')

divert(9)
# add the X-Real-To: header field to the message
# if the mailer is marked with the `T' flag
H?T?X-Real-To: $u
divert(0)
```

After these updates are applied, make sure that sendmail delivers all mail for your domain to one account on the sendmail system. The sendmail configuration for that unified account should list the 'mailer' marked with the 'T' flag.

# LIST Module

The CommuniGate Pro LIST module implements the mailing list mechanism. It also implements mail distribution to [Groups](Groups).

# Mailing Lists

The system administrator can create one or several mailing lists. Users from the same or any other mail system can subscribe to these mailing lists using the Web interface or by sending E-mail. They can post messages on mailing lists by sending E-mail to the list addresses, and posted messages are delivered to all subscribers. All posted messages are stored in the mailing list Mailbox that serves as an archive.

If a user subscribes in the FEED mode, all posted messages are redirected to that user immediately after they are received by the LIST module.

When a user subscribes in the DIGEST mode, the user starts to receive list digests: a multi-part messages generated with the LIST module for each mailing list. Each digest message contains all messages posted on the list since the last digest was generated, prefixed with an index of these messages.

When a user subscribes in the INDEX mode, the user starts to receive messages containing the indexes of newly

posted messages. If the user wants to read some of the posted messages, they can use a Web browser to access the mailing list archive.

---

# Configuring the LIST module

Use the WebAdmin Interface to configure the LIST module. Open the Mail pages in the Settings realm, then open the LIST page.

| Processing | |
|---|---|
| Log Level:   Problems | Processors:   2 |

Log

Use the Log setting to specify what kind of information the LIST module should put in the Server Log. Usually you should use the `Major` (message processing reports) level. But when you experience problems with the LIST module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in these cases more details will be recorded in the System Log. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.

The LIST module records in the System Log are marked with the `LIST` tag.

Processors

When you specify a non-zero value for the `Processors` setting, the LIST module starts to process queued messages directed to mailing lists, starts to generate digests, and starts to clean the mailing list archives. The module can use several simultaneous processors (threads) to process several mailing lists at the same time. If you have more than 50 mailing lists, or if you have many lists with extremely large (10,000+) subscriber lists, you should allow the module to use more than 1 processor.

| Mailing Lists | | |
|---|---|---|
| **Name** | **Domain** | **Owner** |
| SIMS | node5.communigate.com | ali |
| CGatePro | test.communigate.com | kwa |

This table shows all mailing lists created. By following the links in the table, you can open an individual list setup page, the Domain Settings page for the list Domain, or the Account Settings page for the mailing list owner.

## Creating Mailing Lists

Each Mailing List is created inside the Main or one of the secondary Domains, and each Mailing List belongs to its owner - an Account in the same Domain.

To create a Mailing List, create an Account or choose an existing one - the List owner Account. Use the WebAdmin Interface to open the Account Settings page:

| Mailing Lists | |
|---|---|
| **Name** | **Subscribers** |
| rd-list | 1025 |
| marketing | 127 |
| | |

To create a mailing list, type the list name and click the Create Mailing List button.

If you are a Domain Administrator, you should have the Can Create Mailing Lists Access Right to create Mailing Lists in your Domain.

The Server checks that there is no Account or other Object with the same name in this Domain, and creates a new Mailing List.

Several Mailboxes are created in the List owner Account:

| | |
|---|---|
| *listname* | this Mailbox is the Mailing List archive: it contains the messages posted to this Mailing List |
| *listname*/requests | this Mailbox contains the messages with subscription requests. |
| *listname*/reports | this Mailbox contains bounce and other DSN (Delivery Status Notification) messages generated for the messages distributed via this Mailing List. |
| *listname*/approval | this Mailbox contains postings that require the list owner approval (moderated postings). To post these messages, the Mailing List owner should redirect them back to the Mailing List using an authenticated submit method: a MAPI or XIMSS client, the CommuniGate Pro WebUser Interface, the XTND XMIT POP3 method, a local "mail" command, the PIPE module, etc. |

# Configuring Mailing Lists

To configure a Mailing List, open the Mailing List Settings page. You should select a link to the Settings page either from the list of all mailing lists located in the LIST module Settings page, or from the list owner Account Settings page.

You should have the All Domains Server Administrator access right or the Domain Administrator access right in order to open the Mailing List Settings pages via the WebAdmin Interface.

In order to modify a Mailing List settings or subscriber lists, the Domain Administrator should have the CanAccessLists access right.

The list owners can access the Mailing List settings pages using the WebUser Interface with their Accounts. The WebUser session page listing Mailboxes and Mailbox folders also provides links to the Account Mailing Lists pages.

| Log: | All Info | **Subscribers** | Owner: | **listmaster** |
|---|---|---|---|---|

Log

Use the Log setting to specify what kind of information about this mailing list should be put in the Server Log. Usually you should use the Major (message posting, subscription, digest, and clean-up operations) level. But when you experience problems with this particular mailing list, you may want to set the list Log Level setting to Low-Level or All Info: in this case more details will be recorded in the System Log. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.

The mailing list records placed in the System Log are marked with the List(*listname*) tag.

Only the system administrator can change the mailing list Log setting.

| | |
|---|---|
| **Description:** | |
| **Preferred Character Set:** | ISO-8859-1 |
| **Digest & Archive:** | Enabled |
| **Verify Owner Using:** | Authentication |

**Description**

Use the Description setting to specify the full name of the list. It will be used as a "comment" in the list E-mail address as the Real Name setting is used in account E-mail addresses.

**Preferred Character Set**

This option specifies how the List module should handle non-ASCII texts. It is used when displaying list messages via the Web interface: messages in the list can be composed using different character sets, and to display them all on one page, the module should know which character set is preferred.

**Digesting and Archiving**

Posted messages can be stored in a Mailbox created in the list owner Account. Such a Mailbox is used to collect messages for message digests. If messages are not removed from that Mailbox after a digest is created, this Mailbox can be used as a mailing list archive.

**Enabled**

All posted messages are stored in the owner Account Mailbox. Use the link next to the pop-up menu to open the Digesting and Archiving Settings page.

**Disabled**

The posted messages are not stored in the owner Account Mailbox, archiving and digesting options are disabled.
All digest and index-mode subscribers are processed as feed-mode subscribers.
All new attempts to subscribe to the mailing list in the digest or index mode are rejected.

**Verify Owner**

When the LIST module receives an E-mail message, it checks if the message is sent by the List Owner. First, the message Return-Path is compared to the list owner E-mail address. The Return-Path should be *ownerName@listdomain*, where the ownerName is the owner Account name (not one of its aliases), and the listdomain is the list and owner Account Domain name (not one of its Domain aliases).

The Verify Owner setting specifies the additional checks to be made:

**Return-Path**

When this option is selected, no additional check is made.

**IP Addresses**

When this option is selected, the LIST module checks that the message has been submitted either using one of the *authenticating* methods (see below), or via SMTP, from a computer with an IP address included into the [Client IP Addresses](#) list.

**Authentication**

When this option is selected, the LIST module checks that the message has been submitted using one of the *authenticating* methods:

- via [SMTP](#), using the AUTH authenticaiton;
- via [WebUser Interface](#);
- via [POP](#) with XTND XMIT extension;
- via the [PIPE](#) module.

The Settings page contains a set of options, settings, and text areas that controls the subscription, postings, message

distribution, and bounce processing. See the sections below for the details.

## Renaming Mailing Lists

To rename a mailing list, open that list Settings page via the System Administrator Web interface (see above).

You should have the [All Domains](#) Server Administrator access right or the [Domain Administrator](#) access right in order to open the Mailing List Settings pages.

Enter the new name for the list in the Mailing List Settings page, and click the Rename button. The Server checks if there is no Account or Mailing List with the new name in the same Domain and renames the mailing list. Then the Mailing List Settings page is reopened.

## Removing Mailing Lists

To remove a mailing list, open that list Settings page via the System Administrator Web interface (see above).

Click the Remove List button. A confirmation page appears. If you click the Remove button on the confirmation page, the list is removed, the files with the list subscribers and list settings are deleted, and the mailing list Mailboxes are removed from the list owner Account.

# Composing Service Texts

Several Mailing list settings specify texts to be sent to the subscribers - message subjects, message headers and trailers, confirmation requests, etc. A specified text can contain special symbol combinations to be substituted with actual data before the text is inserted into a message.

The following symbol combinations can be used:

| Where | Combination | Substituted with |
|---|---|---|
| | `^N` | the *listname* string |
| All Texts | `^D` | the *domain* string |
| | `^E` | the Description setting |
| Digest Subject, Header, Trailer | `^X` | the sequence number of the current digest |
| Feed Subject Prefix, | `^B` | the sequence number of the current message |
| Header, Trailer | `^C` | the original message 'From' address |

If there is a number after a special symbol combination (as in `^N80`), the number specifies the maximum length of the substitution string.
If a substitution is longer than specified, its last symbols are cut off.

If there is a number after a special string combination, and the number starts with 0, as in `^N040`, the number specifies the length of the substitution string.
If a substitution string is longer than specified, its last symbols are cut off, and if a string is shorter than specified, space symbols are added to the beginning of the string.
If the number starts with two 0 symbols, the 0 symbols are used instead of spaces.

# Subscription Processing

Each mailing list is a list of subscribers, i.e. a list of E-mail addresses receiving messages posted on the mailing list.

In order to subscribe, unsubscribe, and change their [subscription mode](#) via E-mail, users of the *listname* @ *domain* mailing list should send any message to the following addresses:

| Send to address: | New user | Existing subscriber |
|---|---|---|
| `listname-on`@*domain* or `listname-subscribe`@*domain* | to subscribe to the list in the default mode | to confirm the current subscription mode |
| `listname-feed`@*domain* | to subscribe to the list in the FEED mode | to change the subscription mode to FEED |
| `listname-digest`@*domain* | to subscribe to the list in the DIGEST mode | to change the subscription mode to DIGEST |
| `listname-index`@*domain* | to subscribe to the list in the INDEX mode | to change the subscription mode to INDEX |
| `listname-null`@*domain* | to subscribe to the list in the NULL mode | to change the subscription mode to NULL |
| `listname-off`@*domain* or `listname-unsubscribe`@*domain* | | to unsubscribe from the list |
| `listname-confirm`@*domain* | | to get the confirmation ID; this ID can be used as the password for other subscription operations and for list archive browsing |

When subscription request messages are sent to these addresses, the message `From:` header fields are used as requesters E-mail addresses.

You can specify who can subscribe to the mailing list, and how they should subscribe.

| Subscription Policy | | | |
|---|---|---|---|
| Subscribe: | anybody | Requests to Save | accepted |
| Default Mode: | feed | | Request Confirmation |
| **Confirmation Request Message** | | | |
| Subject: | | | |
| Text: | | | |

Subscribe

`nobody`    users cannot subscribe to this mailing list and/or change the subscription mode by themselves. Only the system administrator and the list owner can do these operations. Users still can unsubscribe by themselves.

`this domain only`    only users from the mailing list Domain can subscribe.

`locals Only`    only users registered with this Server can subscribe.

`anybody`    any user on any system can subscribe to this list.

`moderated`    all subscription requests will be stored in the `listname/requests` Mailbox (see below) and should be approved by the list owner. The List owner should redirect these requests to the *listname*-on,

*listname*-feed and other addresses to subscribe new users and to let them change the subscription mode. Users still can unsubscribe by themselves. When user requests are stored in the `listname/requests` Mailbox for approval, they are "flagged" (get the Flag marker).

Save Requests
>This setting specifies which subscription requests (i.e. messages sent to the *listname*-on, *listname*-off and other addresses listed above) should be stored in the `listname/requests` Mailbox in the list owner Account. Messages stored in this Mailbox can be examined if some user reports problems when trying to subscribe to, unsubscribe from, or change the subscription mode to this list.
>You can specify if none, all, only the accepted, or only the rejected request messages should be stored in the *listname*/requests Mailbox.
>Requests waiting for the list owner approval are always stored in the `listname/requests` Mailbox.
>
>All stored request messages get the `X-List-Report` additional header field . This field contains the list manager report (Delivery Status Notification message). It is recommended that the List owners configure their mailer applications so they display the X-List-Report fields.

Default Mode
>When a new user sends a subscription request in to the *listname*-on@*domain* or *listname*-subscribe@*domain* address, i.e. when the subscription mode is not specified, the mode specified with the Default Mode setting is used.

# Subscription Modes

Each list user (E-mail address) is subscribed in one of the following modes:

FEED
>In this mode, the subscriber receives list messages as they are posted. See the [FEED Mode Distribution](#) section for more details.

DIGEST
>In this mode, the subscriber periodically receives *digest* messages. A digest message starts with the Table Of Content - the list of the messages posted, followed by the posted messages themselves. See the [DIGEST/INDEX Mode Distribution](#) section for more details.

INDEX
>In this mode, the subscriber periodically receives *index* messages. An index message is the same as the digest Table Of Content, but it does not contain the posted messages themselves. INDEX subscribers can see if they are interested in any posted messages, and use the Web interface to read those messages in the mailing list archive. See the [DIGEST/INDEX Mode Distribution](#) section for more details.

NULL
>In this mode, the subscriber does not receive any messages from the list. This mode can be used by the "posters" - the list users that only post messages on the list.

BANNED
>The "banned" users do not receive messages from the list, and they cannot change their subscription mode themselves.
>You can use this method to make it impossible for certain users to subscribe to your mailing lists, though usually the more generic anti-spam and other system-wide protection methods should be used.

# Confirmation Requests

There are several very common problems with most of publicly available mailing lists:

- subscription messages are sent from incorrect addresses, those incorrect addresses are inserted into the

subscribers list, and mailing list messages are repeatedly sent to those incorrect or unused addresses.
- list abusers (such as spammers) subscribe nonexistent E-mail addresses just to allow themselves to post on the lists that accept posts from subscribers only.
- using easily forged message headers, some persons can subscribe and unsubscribe another person E-mail addresses.

These and some other problems can be solved using confirmation requests.

`Request Confirmations`

When this option is selected, the List manager does not fulfill subscription requests immediately. Instead, a confirmation request message is composed and sent to the address that is about to be included or excluded from the subscribers list. The confirmation request contains a unique identifier (Confirmation ID) in its Subject field. When the user receives such a confirmation request, they can simply use the mailer Reply command to confirm the requested operation.

`Confirmation Message`

These text settings allow you to specify the subject and the body of confirmation request messages the module sends to the subscribers. In addition to generic "symbol combinations", these service texts can also contain the following combinations:

| Combination | Substituted with |
|---|---|
| `^O` | the requested operation |
| `^P` | `unsubscribe` for the `unsubscribe` operation, `subscribe` for the `subscribe` operation, and `subscribe(operation)` for other operations |
| `^A` | the subscriber address |
| `^I` | the confirmation identifier |

# Welcome and Good Bye Messages

| Welcome/Policy Message |
|---|
| Subject: |
| Text: |

| Good Bye Message |
|---|
| Subject: |
| Text: |

When a new user is subscribed, the Policy Text message is sent to that new user. When a user unsubscribes, the Good Bye message is sent.

Besides the generic "symbol combinations", these service texts can also contain the following combinations:

| Combination | Substituted with |
|---|---|
| `^A` | the subscriber address |
| `^I` | the confirmation identifier |

# Posting Messages

To post a message on the mailing list, the author should send it to the *listname@domain* address.

| Posting Policy | | | |
|---|---|---|---|
| Accept Postings: | from subscribers | New Subscribers: | Moderate first 2 |
| Allowed Format: | text only | Maximum Size: | 30 Kbytes |
| Prohibit: | Non-matching Character Set<br>Unmodified Digest Subjects | | |
| Service Fields: | | | |
| Compose 'From' Address as: | Author Name with Address and List Address | | |

Accept Postings
: This setting specifies who can post messages on this mailing list:

`from owner only`
: Only messages submitted by the list owner (using any authenticated method) will be posted

`moderated`
: Messages from everybody but the list owner are redirected to the list owner for approval; if the list owner redirects a message back to the list, the message is posted.
**Note:** this mode can be used to change the list posting policy when the list owner wants to postpone all postings. If you use the `from subscribers` mode, the New Subscribers setting (see below) provides more advanced moderation options.

`from subscribers`
: Messages from the list subscribers are accepted for posting; some messages can be moderated (see below).

`moderate guests`
: Messages from the list subscribers are accepted for posting; some messages can be moderated (see below). Messages from addresses that are not subscribed are moderated.

`from anybody`
: Messages from any address are accepted for posting.

New Subscribers
: This setting is effective only if the Accept Postings setting is set to `from subscribers`. When a user subscribes and starts to post messages, the messages are stored in the list owner Mailbox waiting for approval. After the specified number of messages is approved and posted, all messages sent by this user are posted on the list directly, without the list owner approval.
**Note: this is a very effective way to enforce the list policies and to protect the mailing list from "spamming".**
If you set this option to `Moderate All`, then all messages from new subscribers will be stored for approval (the LIST module will not update the posted messages counter).
If you set this option to `Prohibited` new subscribers won't be able to post on the list (their postings will be rejected).
If you set this option to `Special`, new subscribers will be able to post auto-generated messages. This is useful if you want to subscribe this mailing list to other mailing lists.
**Note:** when an auto-generated message should be posted on the list, and the message is not a message generated by the list owner, the message `Sender` address (if exists) is used instead of the `From` address. If that address is subscribed to the list, and the subscriber posting mode is set to `Special`, the message is posted.
You can open the [subscribers list](#) and change this setting for individual subscribers. You may want to change this setting to `Unmoderated` for some users, letting only those users post messages on the list, while posting from all other users (and new users) will be either stored for approval or rejected.

Allowed Format
: This setting specifies the allowed MIME format for postings.

| | |
|---|---|
| `plain text only` | only messages in the text/plain format can be posted |
| `text only` | only messages in the text (text/plain, text/html, etc.) formats can be posted |
| `text alternative` | only messages in the text format or multipart/alternative messages that contain a part in the text format can be posted (for example, a message can contain a `text/html` variant part and the same text as a `image/gif` variant part) |
| `anything` | messages in any MIME format can be posted |

The list owner can always post messages in any format.

**Maximum Size**
   This setting restricts the size of messages that can be posted on this mailing list. The list owner can always post messages of any size.

**Prohibit Unmodified Digest Subject**
   When this option is selected, the Subject fields of all postings are checked. If the Subject is a "reply prefix" (such as Re:, Re>, etc.) followed by this list Digest String (see [below](#)), the message is rejected.

**Prohibit Non-matched Character Set**
   When this option is selected, the character set used to compose the posting is checked. If the character set is explicitly specified, and it does not match the Preferred Character Set for this list, the message is rejected.

**Service Fields**
   Use this option to specify the message header fields to be added to all distributed messages (feed, digest and index). If this option value starts with the asterisk (`*`) symbol, the module automatically generates the `List-ID`, `List-Unsubscribe`, `Precedence` and (if list browsing is enabled) `List-Archive` fields.
   Otherwise, the specified fields are added. The following symbol combinations can be used in this field:

| Combination | Substituted with |
|---|---|
| `^N` | the *listname* string |
| `^D` | the *domain* string |
| `^P` | the first HTTP User module port |
| `^R` | the `http` or the `https` string, depending on the type of the first HTTP User module port |

**Compose 'From' Address as:**
   This setting specifies how the 'From' addresses of the senders are modified in posted messages.

| | |
|---|---|
| `Author Name and Address` | The original Author name (the comment part) and address are used. |
| `Author Name and List Address` | The Author name is used but the address is replaced with the List address. Use this option to make List messages to be compatible with DMARC. |
| `Author Name with Address and List Address` | Same as above but also the Author address is inserted into the comment part. |

---

# Processing Messages

When a posted message is being sent to subscribers, the original message header is modified. Usually, only the From, Date, Message-ID, and Subject fields are copied from the original message.

You can specify additional header fields to be copied from original postings to the messages sent to subscribers - directly or as parts of digests.

| **RFC822 Fields to Keep** |
|---|
| To/Cc:   remove |

RFC822 Fields to Keep

Use these fields to specify the names of additional RFC822 header fields to be copied from the original postings to the distributed messages. If you want to remove a name, enter an empty string into the name field.

If you want to copy all header fields, enter the asterisk (`*`) symbol into the first field.

If you want to copy the Message-ID for the messages distributed using the "feed" mode, include the `Message-ID` field name.

If the To And Cc option is set to:

- `remove` - the original `To` and `Cc` fields are not copied; the Mailing List address is added as the `To` field;
- `keep` - the original `To` and `Cc` are copied;
- `copy as Cc` - the original `To` and `Cc` fields are copied as `Cc` fields; the Mailing List address is added as the `To` field.

# Bounce Processing

Incorrect and expired E-mail addresses create the most annoying problems for mailing list administrators. The CommuniGate Pro LIST module automates processing of incorrect, expired, and temporarily unavailable addresses.

All messages sent to subscribers (in all modes) have message envelope information that routes all error reports back to the LIST module. When a report is received, the LIST module:

- stores the report in the *listname*/reports Mailbox in the owner Account (optional);
- parses the report text;
- if the report has a correct delivery-report (RFC1892/RFC894) structure, processes all records in the report.

Most of the problems can be detected immediately when sending a list message from the CommuniGate Pro Server, so most of the error reports are generated locally, on the same CommuniGate Pro server. The CommuniGate Pro server generates reports in the proper format, so most of the delivery problems are handled automatically.

If a list message has been sent to a remote site without a problem, but then that remote site fails to deliver the message to the recipient, the delivery report is generated on that remote site. Most of the modern mail servers generate delivery reports in the correct format, so in this case many problems are handled automatically, too.

And, finally, it is still possible to receive unformatted delivery reports from other sites. The LIST module can store those unformatted reports in the *listname*/reports Mailbox, so the list owner can process them manually.

Each record in the delivery report contains information about one E-mail address, and indicates if the original message was or was not delivered to that address. It can also specify if the delivery problem is fatal (as when account is removed from the system), or if it is a non-fatal, temporary problem (as when a remote site is down or when the account disk space quota is exceeded).

If a non-fatal report is received, the E-mail address in question is suspended, and no list messages are sent to that address, and all additional error reports about that address are ignored. When the LIST module performs periodic list clean-ups, it sends a warning message to all suspended addresses. The warning message notifies that user that some list messages have not been delivered to the user address, and it also asks the user to confirm subscription (by replying to the warning message).

There are two ways to specify the suspension period:

- An address can be suspended for a fixed period of time. When that period ends, the LIST module resumes sending list messages to this address. This can result in new bounces which will increment the bounce counter,

and the address is unsubscribed after the bounce counter exceeds the specified limit.
The warning messages are sent to the failed address after the suspension is over, too - till the user confirms the subscription by replying to the warning message.
- An address can be suspended till the user confirms subscription by replying to the warning message. If no confirmation is received during the specified period of time, the address is unsubscribed.

When a user confirms the subscription by replying to the warning message, the suspension period ends, and the bounce counter associated with the user E-mail address is cleared.

| Bounce Processor | | | |
|---|---|---|---|
| On a Non-Fatal Bounce: | suspend subscription for: | 6 hour(s) | |
| | unsubscribe after: | 5 | bounces |
| | suspend till confirmation | | |
| | unsubscribe after: | 7 day(s) | |
| Process a Fatal Bounce as: | Fatal | | |
| | Notify Owner when Unsubscribing | | |
| Cleanup List every: | 24 hour(s) | | |
| Bounce Reports to Save: | unprocessed | | |
| Warning Message | | | |
| Subject: | | | |
| Text: | | | |

suspend subscription for
> If this option is selected, it specifies for how long a subscriber should be suspended if the system receives a non-fatal problem report about the subscriber's E-mail address.
> The `unsubscribe after` option specifies the number of unconfirmed suspension periods after which the user is unsubscribed.

suspend till confirmation
> If this option is selected, a non-fatal error report suspends the address till the subscriber sends a confirmation message.
> The `unsubscribe after` option specifies the time period to wait for a confirmation message.

Process a Fatal Bounce as
> This setting specifies how the system should process fatal problem reports: as non-fatal, as several non-fatal, or as a fatal problem. If you specify that the system should unsubscribe a user after receiving 10 non-fatal problem reports about the user address, and you specify that a fatal problem report should be processed as 5 non-fatal reports, this will tell the system to unsubscribe the user after 2 fatal reports. If you specify that a fatal problem report should be processed as fatal, the system will unsubscribe a user immediately upon receiving a fatal problem report.

Notify Owner When Unsubscribing
> If this option is selected, an E-mail message is sent to the List Owner every time an address is unsubscribed because of mail bouncing.

Cleanup List every
> This setting specifies how often the system should scan the subscription list. When scanning the list, the system:

- sends warning messages to the subscribers with non-zero bounce counter;

removes subscribers who sent subscription requests more than 2 days ago and who have not confirmed the subscription requests;

- removes unsubscribed addresses from the list.

---

`Bounce Reports to Save`
This setting specifies which delivery reports should be saved in the `listname/reports` Mailbox in the list owner Account. If you specify `unprocessed`, only the messages that the LIST module fails to parse and process are stored in that Mailbox.

`Warning Message`
These text settings specify the subject and the text of a warning message that is sent to subscribers when their subscription is suspended. In addition to generic "symbol combinations", this service text can contain the following combinations:

**Combination  Substituted with**

`^A`           the subscriber address

`^I`           the confirmation identifier

---

# FEED Mode Distribution

After a message is posted, it is distributed to all users subscribed in the FEED mode.

The `To` header field of a distributed message contains the mailing list address. The `From`, `Date`, and `Message-ID` fields (and specified [additional fields](#)) are copied from the original posting.

The body of the distributed message is a copy of the original message body. If the original message was not in the MIME format, or if it was in the MIME text/plain or multipart/mixed format (the most common formats), the FEED Mode Header text is inserted before and the FEED Mode Trailer text is inserted after the body of the original posting.

| Feed Mode Format | |
|---|---|
| Subject Prefix: | Direct Replies:   to List |
| insert after Reply Prefix | |
| Header | |
| Trailer | |
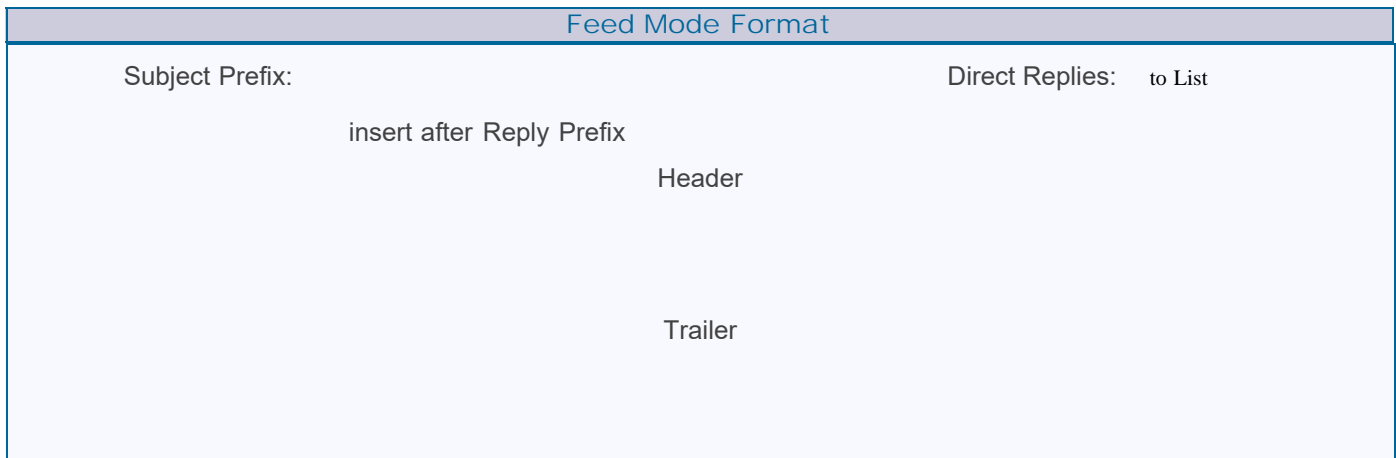
`Subject Prefix`
This setting specifies the string that is inserted into the beginning of the Subject field of all messages distributed in the FEED mode.
When a message is distributed, the system checks the Subject field. If the subject prefix found after the reply prefix (`Re:`, `Re>`, etc.), then the subject prefix is deleted.
The Subject Prefix string can contain special [symbol combinations](#).

`insert after Reply Prefix`
Select this option to insert the Subject Prefix after the Reply prefix (this helps client mailers to group related messages into *discussion threads*).

550

Sample:

| the Subject Prefix setting | `[R&D]` |
| a posted message | `Subject: test` |
| the distributed message | `Subject: [R&D] test` |
| the posted reply | `Subject: Re: [R&D] test` |
| the distributed reply<br>(`insert after Reply Prefix` is not selected) | `Subject: [R&D] Re: test` |
| the distributed reply<br>(`insert after Reply Prefix` is selected) | `Subject: Re: [R&D] test` |
| the composed digest | `1) test`<br>`2) Re: test` |

`Direct Replies`

If the `to List` option is selected, the Reply-To header field with the E-mail address of the list is added to all distributed messages. As a result, when subscribers answer to distributed messages, their replies are directed to the mailing list by default.

If the `to Sender` option is selected, the Reply-To header is inserted with the From address of the message author (sender).

`Header`

This setting specifies the text to be included in the beginning of messages distributed in the FEED mode.
The Header string can contain special [symbol combinations](#).

`Trailer`

This setting specifies the text to be included at the end of messages distributed in the FEED mode.
The Trailer string can contain special [symbol combinations](#).

# Digesting and Archiving

Posted messages can be stored in a Mailbox created in the list owner Account. Such a Mailbox is used to collect messages for message digests. If messages are not removed from that Mailbox after a digest is created, this Mailbox can be used as a mailing list archive.

Select the `Enabled` value for the Digesting and Archiving option and follow the link next to that setting to modify the Digesting and Archiving settings.

# DIGEST/INDEX Mode Distribution

When the Digesting and Archiving option is enabled, all posted messages are stored in a Mailbox created in the list owner Account. The LIST module periodically checks that Mailbox and creates list digests and indexes.

A digest message contains a set of the messages posted on the mailing list since the time when the previous digest was composed.

A digest message body contains the digest header, the table of contents (TOC) listing all the messages in the digest, the TOC trailer, the posted messages, and the digest trailer.

A list index message contains the same digest header, TOC, and TOC trailer, but it does not contain the posted

messages themselves and it does not contain the digest trailer.

List index messages are created at the same time the list digest messages are created. Digest messages are sent to the digest-mode subscribers, and index messages are sent to the index-mode subscribers.

| Digest Generator | | |
|---|---|---|
| Generate every: 24 hour(s) | | |
| or if Larger than: 100K | or when messages collected: | 100 |
| First Digest at: 5:00AM | | |

`Generate Every`
> This setting specifies how often the LIST module should generate digest (and index) messages for this list.

`First Digest at`
> This setting specifies the time of the day when the first digest should be generated.
>
> **Note:** if the Generate Every setting value is not more than 1 day, every day the first digest is generated at the specified time. If this setting is set to 4:00, the Generate Every setting is set to 1 Day, and the last digest was generated at 23:00 on Monday, the next digest will be generated at 4:00 on Tuesday.
> If the the Generate Every setting value is set to N days (N>1), the first digest is generated at the specified time N days later: if this setting is set to 4:00, the Generate Every setting is set to 5 Days, and the last digest was generated at 23:00 on Monday, the next digest will be generated at 4:00 on Friday.

`if Larger than`
> This setting specifies the maximum size of the messages to be included into one digest. If the total size of all messages posted since the last digest was generated exceeds this limit, a new digest (and index) is generated immediately, overriding the `Generate Every` and `First Digest at` settings.

`if has X messages`
> This setting specifies the maximum number of messages to be included into one digest. As soon as the specified number of messages is posted, a new digest is generated immediately.

| Digest Format | |
|---|---|
| Subject: | Body Format: plain text |
| **Header** | |
| **Index Line** | |
| **Index Trailer** | |
| **Trailer** | |

Digest Subject
>This text setting specifies the Subject header field for digest and index messages created for this mailing list.
>If the [Prohibit Unmodified Digest Subjects](#) option is selected, the mailing list manager rejects all postings with a reply prefix (`Re:`, `Re>`, etc.) followed by an unmodified Digest Subject text.

Body Format
>This setting specifies how the digest body should be formatted.

>plain text
>>Digests are composed as plain text messages; individual messages are separated with a line containing minus symbols.

>standard MIME
>>Digests are composed in the MIME multipart/digest format, where the first part has the text/plain Content-type and contains the digest TOC, and other parts contain the messages (postings).

>embedded MIME
>>Digests are composed in the MIME multipart/mixed format, where the first part has the text/plain Content-type and contains the digest TOC, and the second part uses the standard multipart/digest format and contains all messages (postings).

Header
>This text setting specifies the text to be included into all digest and index messages before the Index Lines (Table of Contents).

Index Line
>This setting specifies the format for an index entry. Like the service text settings, the Index Line can (and should) have special symbol combination, but these combinations are different.
>For each posted message, values from the message are substituted into the Index Line text and the resulting line is stored in the digest or index message being composed.

| Combination | Substituted with |
|---|---|
| ^X | the message sequence number in the digest being composed |
| ^F | the `From` header field of the message |
| ^T | the `Date` header field of the message |
| ^S | the `Subject` header field of the message |
| ^I | the `Message-Id` header of the message |

Index Trailer
>This text setting specifies the text to be included into all digest and index messages after the Index Lines.

Trailer
>This text setting specifies the text to be included after the last message in the digest.

# Archiving

All messages posted on a *listname* mailing list are stored in the *listname* Mailbox in the list owner Account. When a digest is being composed, the posted messages are retrieved from that Mailbox. This Mailbox also serves as an archive for all posted messages, and this archive can be searched via the user Web interface.

Each time after a digest is composed, the Mailbox is checked and the oldest posted messages are removed to keep the archive Mailbox size within the specified limits.

| Archiving | Enabled | | |
|---|---|---|---|
| Maximum Archive Size: | 300 Kbytes | Messages to Keep: | 1000 |
| Start new Archive every: | month | Browse Access: | subscribers |

Maximum Archive Size
> This setting specifies the maximum size of the archive Mailbox. After a digest is generated, the new archive file can be generated or the oldest messages can be removed from the Mailbox to keep the Mailbox size below the specified limit.

Messages to Keep
> This setting specifies the maximum number of messages that can be left in the archive Mailbox after a digest is generated.

Start New Archive
> This setting specifies when the new archive Mailbox should be created. The old archive Mailbox becomes a submailbox with the name YYYY-MM-DD, where YYYY specifies the year, MM specifies the month, and DD specifies the day when the first archive message was stored.
> Unless the Start New Archive option is set to `never`, a new archive is created when the archive Mailbox size limit or the archived message number limit is exceeded.
> If the Start New Archive option is set to `never` and the Maximum Archive Size option is set to zero, all messages are removed from the archive Mailbox as soon as a digest is generated.

Who can Browse
> This setting specifies if this mailing list should appear in the Mailing Lists section of the WebUser Interface.

> nobody
> > The mailing list will not be available via the Web Interface.

> anybody
> > The mailing list will be displayed as a browsable mailing list, and its archive can be used by anybody.

> subscribers
> > The mailing list will be displayed as a browsable mailing list, but in order to browse its archive, users should enter their E-mail address and the Confirmation ID (as the password). To retrieve a forgotten Confirmation ID, a user can always send a message to `<listname-confirm@domain>` and get the confirmation ID even if the list subscription mode does not require confirmations.
> > If the subscriber (E-mail address) is a local CommuniGate Pro Account, then not only the confirmation ID, but also the Account password can be used as the list access password.

> clients
> > The mailing list will be displayed as a browsable mailing list, and can be browsed from the Internet addresses included into the server [Client IP Addresses](#) list.
> > For users trying to view the list from outside the specified addresses/networks this mode works as the `subscribers` mode: these users have to enter the name/password pair (E-mail and Confirmaton ID) to browse the list.

# Subscribers List

If you are a system administrator or the list owner, you can access the subscribers list page following the [Subscribers](#) link on the Mailing List Settings page.

The Subscribers page contains the list of all E-mail addresses subscribed to the mailing list. For each address additional information (such as the subscriber's real name, number of bounces from this address, etc.) is listed. Each address can be marked, and you can use the Mark All button to mark all list subscribers. You can use the Filter field to display the subscribers with matching addresses only.

| 30 | Filter: | | | | | 2 of 2 selected |
|---|---|---|---|---|---|---|
| **E-mail** | | **Mode** | **Subscribed** | **Posts Bounces** | **Real Name** | |
| andy@vax.communigate.com | | null | 15:54:51 | 3 | Andy | |
| test@mail.communigate.com | | feed | 18:18:11 | 2 mod | Test Account | |
| | | | Unmoderated | | | |
| | | | feed | | | |

Unsubscribe
> Mark some subscribers and click this button to unsubscribe them from the list. Depending on the current FeedBack setting value, the LIST module will either unsubscribe them immediately or just send them confirmation requests. If the FeedBack setting (see below) value is `Send Welcome`, the `Good Bye` messages are sent to unsubscribed addresses.

Mark Failed
> Mark some subscribers and click this button to tell the LIST module that mail to those addresses bounced. This can be useful in situations when the LIST module fails to process bounce reports automatically, because they come in a non-standard format. Clicking the Mark Failed button will result in the same actions (increased bounce counter, suspension, and warning generation) as caused by receiving a non-fatal bounce from the marked address.

Set Postings
> These controls allow you to change the moderation mode for the selected users. You can select some subscribers and set their posting mode to moderated, prohibited, unmoderated, or special. See the Posting Messages section for more details.

Set Mode
> These controls allow you to change the subscription mode for the selected users. See the Subscription Processing section for more details. If the FeedBack setting (see below) value is set to `ask Confirmation`, the subscription mode is not changed, but a confirmation request for the mode change operation is sent to the marked subscribers.

## Adding Subscribers

You can manually add subscribers to the list. Enter the new subscriber E-mail address press the Subscribe button.

| Single Address: | | Feedback |
|---|---|---|
| Import: | no file selected | silently |

Feedback
> If this option is set to `ask Confirmation`, all operations performed result in confirmation requests being sent to the specified subscriber address.
> If this option is set to `Send Welcome`, confirmation requests are not generated, and the subscription modes are changed immediately, but the Welcome and Good Bye messages are sent to subscribers when you unsubscribe a user or subscribe a new user.
> If this option is set to `silently`, no messages are sent to subscribers.

`Single User`
Use this field to enter an E-mail address of the user you want to subscribe to this list. A new subscriber address can be specified as an E-mail address with a comment, as `John Smith <johns@company.com>` or `johns@company.com (John Smith)`, in this case the comment is stored the subscriber's real name.
Click the Subscribe button to subscribe the specified address.

## Importing Subscriber Lists

You can use text files with E-mail addresses to add subscribers to mailing lists. Open the [Subscribers](#) page, and use the Import control to select a file with E-mail addresses. Click the Subscribe button to add the addresses to the mailing list.

The text file should have one E-mail address per line, with several optional fields on each line. If a line contains several fields, they should be separated with the tabulation (TAB) symbol.

- The first and the only required field is the E-mail address.
- The second field specifies the subscription mode. The first field symbol is checked. The symbols `d` and `D` require the DIGEST mode, the symbols `I` and `i` - the INDEX mode, and the letters `f` and `F` - the FEED mode. All other field symbols are ignored. If the first symbol is not recognized or the field is absent, the new user is subscribed in the Mailing List default mode.
- The last field (if a line contains more than 2 fields) specifies the real name of the new user.

The Mailing List manager checks the file format first. If the file format is incorrect, no new user is subscribed. This allows you to fix the file format and to try the same file: either all addresses are added, or none is added.

**Note:** The import file must be prepared on the client computer (on the computer you use to run your browser). The browser allows you to upload files from disks connected to that computer, not to the CommuniGate Pro Server computer.

**Note:** When using Netscape and some other Unix browsers, make sure that the file name ends with the `.txt` suffix - otherwise the browser won't upload the file as a text one, and the file will be ignored.

**Note:** Some versions of the Netscape® browser for "classic" MacOS® do not convert the MacOS text files (that use the CR symbol as the line separator) into CR-LF delimited text files. You may see the "format error" messages if you try to import a subscriber list from a MacOS computer using that browser. You should either use a different browser, or you should convert the subscriber list into a CR-LF delimited text file before importing it with that browser.

**Note:** If you are moving users from a different mailing list system, make sure you have set the Feedback option to Silently - otherwise all inserted subscribers will receive confirmation requests and/or Welcome messages.

## Subscribing Lists to Lists

You may want to subscribe List1 to List2, so all messages posted on the List2 list are sent to the List1 subscribers, too.

After you have added the qualified address of the List1 (list1@domain1.dom) to the List2 subscribers list, add the qualified address of the List2 (list2@domain2.dom) to the List1 susbcribers list. Set the subscription mode to `null`, so messages will not go back to the List2 list, and set the posting mode to `special`, so messages from List2 (which are auto-generated list messages) will be allowed for posting on List1.

---

# Processing Service Requests

The LIST module processes messages sent to `listserver@localdomainname`. The module takes the List Server

commands from the message body, processes those commands, and composes a response message with the command execution results.

All commands sent to the above address apply to the mailing lists in the specified Domain only (to the virtual list server in that Domain).

The messages with the List Server commands should be in the plain text format, or in the mulitpart/alternative format containing a part in the plain text format.

Each List Server command is stored on one text line. Line starting with the `%`, `*`, `#`, and `;` symbols are not processed (comment lines).

The following commands are supported:

```
SUBSCRIBE listname [mode [confirmation ID]]
SUB listname [mode [confirmation ID]]
```
> These commands subscribe the message author to the *listname* mailing list. If the *mode* is not specified, the default subscription mode is used.
> This is equivalent to sending a message to the `listname-on@localdomainname` address.

```
UNSUBSCRIBE listname [confirmation ID]
UNSUB listname [confirmation ID]
```
> These commands unsubscribe the message author from the *listname* mailing list.
> This is equivalent to sending a message to the `listname-off@localdomainname` address.

```
CONFIRM listname
GETID listname
```
> These commands send the message author his/her *listname* subscription ConfirmationID (password).
> This is equivalent to sending a message to the `listname-confirm@localdomainname` address.

```
WHICH
CHECK
```
> These commands list the mailing lists (in this *localdomainname* Domain) the message author is subscribed to.

```
HELP
```
> This command displays the list of supported commands.

```
QUIT
FINISH
```
> These commands tell the LIST module to stop processing. The rest of the message text is ignored.

---

# Routing

The LIST module routes to itself all following addresses:
`listname@domain`
if the list `listname` exists in the `domain` Domain, or
`listname`
addresses, if the mailing list `listname` exists in the Main Domain.

Messages to these addresses are processed as submissions to the specified mailing lists.

The LIST module detects addresses in the form
`listname-request@domain` and `listname-admin@domain`
or
`listname-request` and `listname-admin`.

Messages sent to these addresses are rerouted to the mailing list owner.

The LIST module also routes to itself the following addresses:
*listname-xxx@domain*
and
*listname-xxx.*

The xxx suffix can be one of the following:

| suffix | Action |
| --- | --- |
| `on`, `off`, `subscribe`, `unsubscribe`, `feed`, `digest`, `index` | messages sent to these addresses are processed as [subscription requests](#). |
| `report` | messages sent to these addresses are processed as delivery reports about the distributed messages |
| anything else | messages are rejected |

Version 6.2

Overview | Rules | Filters | SMTP | Local | RPOP | LIST | **PIPE**

# PIPE Module

- **The Submitted Folder**
- **Delivering to External Applications**
  - **Serialized Delivery**
  - **Command Tags**
- **Configuring the PIPE module**
- **Foreign Queue Processing**

The PIPE module allows external applications running on the Server computer to submit messages to the CommuniGate Pro Server bypassing TCP/IP connections and Internet protocols, and it allows the Server to deliver messages to external applications.

The PIPE module is also used to submit messages composed with the `mail` and `sendmail` programs that come with the CommuniGate Pro server software. These programs are designed as drop-in substitutions for legacy `mail` and `sendmail` programs.

# The Submitted Folder

The CommuniGate Pro PIPE module creates the `Submitted` folder inside the CommuniGate Pro *base folder*.

The PIPE module scans this folder periodically, and processes the files with the `.sub` file name extension. When such a file is found, the module copies it into a *message queue* file and submits that file to the Server kernel for processing.

The `.sub` text files should contain messages in the RFC822 format. The module uses the data in the RFC822 header fields to compose the message envelope.

- The RFC821 "channel format" should NOT be used: submitted files should use the system native "End of Line" character(s), the dot (.) symbol in the first position should not be doubled, and the file should not end with the dot (.) symbol.
- Addresses specified in the `To:`, `Cc:`, `Bcc:` header fields are used to create the message envelope (recipient addresses).
- The `Bcc:` header fields are removed from the submitted message.
- If at least one `Envelope-To:` header field is detected, message envelope (recipient) addresses are formed using the addresses specified in these header fields, and addresses in all **remaining** To, Cc, and Bcc headers are not placed into the message envelope. The `Envelope-To:` header fields are removed from the submitted message.
- If no `Envelope-To:` header field exists, and the `Envelope-Ignore` field or fields exist, the addresses specified in

559

To/Cc/Bcc header fields and also listed in the Envelope-Ignore fields are NOT included into the message envelope.

- If the `Sender:` and/or `From:` header fields contain addresses without the domain part, the Server domain name is added to those addresses.
- If the `Return-Path:` header field exists, the address specified in that header is used to compose the Return-Path envelope address, and this header field is removed from the submitted message.
- If the `Return-Path:` header field does not exist, the address specified in the `From:` or `Sender:` header field is used to compose the envelope Return-Path address.
- If the `Envelope-ID:` header field exists, its content is used as the message Envelope ID. This header field is removed from the submitted message.
- If the `Envelope-Notify:` header field exists, it should contain one or several keywords `SUCCESS,FAILURE,DELAY,RELAY` separated with comma symbols, or it should contain one keyword `NEVER`. This header field specifies the DSN parameters applied to all envelope addresses composed after this field is met. This header field is removed from the submitted message.
  Several `Envelope-Notify:` header fields can be used to specify different DSN parameters for different addresses.

If processing of a `.sub` file fails (for example, if the file does not have any recipient address), the module places a record into the System Log, and changes the file extension to `.bad`.

If a .sub file is submitted successfully, the file is deleted from the Submitted folder.

Because of the way the PIPE module processes the Submitted folder, it is recommended to compose messages in a different file directory and then move the composed .sub files to the Submitted folder, or to compose messages in the Submitted folder, in files with the `.tmp` file name extension, and then change the file name extension to `.sub`.

Messages submitted via the PIPE module are marked as "received from a trusted source", so they can be relayed without restrictions.

The Submitted folder is used for [Legacy Mail Emulation](#).

# Delivering to External Applications

The PIPE module accepts all messages directed to the `pipe` domain.

The local part of the message address specifies the external application to launch. The part can contain parameters, and can be enclosed into the quotation marks.

Example:
A message directed to the `"execjoe -l store"@pipe` address will be sent to the application `execjoe` started with the `-l store` parameters.

You usually use the PIPE delivery via the [Router](#):
```
<*@somedomain> = exec*@pipe
```
this Router record will direct messages sent to the `joe@somedomain` address to the `execjoe` application.

```
<*@somedomain> = "execall\ -u\ *"@pipe
```
this Router record will direct messages sent to the `joe@somedomain` address to the `execall` application started with the `-u joe` parameters.

To limit the set of applications that can be started via the PIPE module, the *external application directory* is specified as one of the PIPE module settings. The application names specified in message addresses can not include the slash (`/`) or the backslash (`\`) symbols, and they cannot start with the dot (`.`) symbol, and it specified the name of the

application (program) file in the *external application directory*.

The message text (including the message headers and the message body) is passed to the external application as its *standard input*.
**Note:** the application must read the entire *stdin* data stream, otherwise message processing fails.

When the external application completes, the PIPE module reads and discards the application *standard output*. Make sure that your application does not write anything to its *standard output*, so it is not blocked when the communication channel (pipe) buffer between the application and the Server is full.

When the external application completes, the PIPE module reads its *standard error* channel. If it is not empty, the message delivery fails, and the text written to *standard error* is sent as an error report to the message sender.

## Serialized Delivery

In order to allow several PIPE processors to deliver messages simultaneously, the PIPE module creates a separate queue for each message it has to deliver. If you want to serialize processing, you can use the following form of the PIPE address:

`"queue[`*name*`] application parameters"@pipe`

All messages directed to these addresses will be placed into the *name* queue, and a single PIPE processor will send the enqueued messages to the application(s) specified in those addresses. You can use any alphanumeric string as a queue *name*, and you can specify as many queues as you need.

The following [Router](#) records can be used to maintain two serialized PIPE queues (the PROC1 and ARCH queues):

```
<incoming> = "queue[PROC1] procin -mark"@pipe
<control>  = "queue[PROC1] procin1 -control"@pipe
<archiver> = "queue[ARCH] appendfile /var/archive"@pipe
```

All messages sent to the <incoming@maindomain.com> and <control@maindomain.com> will be processed one-by-one using one PIPE processor.

For PIPE addresses that do not have the `queue[`*name*`]` prefix, the PIPE module creates separate queues with numeric names.

## Command Tags

The message text (the header and the body) is sent to the task as that task *standard input (stdin)*.

An application name can be prefixed with the `[FILE]` tag:

`[FILE] application parameters`

When this prefix is used, the application standard input will be empty (closed), or it will contain only the message header fields added by Rules. The string `-f Queue/`*fileid*`.msg` (the -f flag and the Message file name, relative to the *base directory*) will be appended to the end of the application parameters:

`-f Queue/12002345.msg`

**Note:** The beginning of a Queue file contains the service information (envelope, options, etc.). The application should ignore this information skipping the file data till the first empty line. The message itself starts after that first empty line in the Queue file.

**Note:** Header fields added to the message by Server-wide and Cluster-wide [Rules](#) are not stored in the message Queue file, they are sent via the task *standard input*.

**Note:** this prefix should not be used on MS Windows and IBM OS/2 platforms, as the Server keeps the message file open, making it impossible for an external Task to read the file.

An application name can be prefixed with the `[RETPATH]` tag:

`[RETPATH] application parameters`

When this prefix is specified, the string "-p" followed by the message return-path address is added to the end of the application parameters:

`-p address@domain.com`

An application name can be prefixed with the `[RCPT]` tag:

`[RCPT]application parameters`

When this prefix is specified the string "-r" followed by the original recipient address is added to the end of the application parameters:

`-r address1@domain1.com`

An application name can be prefixed with the `[STDERR]` tag (see below).

An application name can have several prefix strings, and they can be specified in any order. If several of `[FILE]`, `[RETPATH]`, and `[RCPT]` prefix strings are specified, the `-f` flag and its parameter are added first, followed with the `-p` flag and its parameter, followed with the `-r` flag and its parameter.

If the `[STDERR]` prefix is specified and the external application completes sending some data to its *standard error* channel, the *standard error* data is used to compose the error report text.

---

# Configuring the PIPE module

Use the WebAdmin Interface to configure the PIPE module. Open the Mail pages in the Settings realm, then open the PIPE page.

| Processing | |
|---|---|
| Log Level: Problems | Processors: 3 |
| Application Directory: | |
| Processing Time Limit: 2 minutes | |

`Log`
> Use the Log setting to specify what kind of information the PIPE module should put in the Server Log. Usually you should use the `Major` (message transfer reports) or `Problems` (message transfer and non-fatal errors) levels. But when you experience problems with the PIPE module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.
>
> The PIPE module records in the System Log are marked with the `PIPE` tag.

`Processors`
> This option specifies the number of threads used to deliver messages. If some of your external applications are slow, you may want to use several PIPE delivery threads, so several messages can be processed at the same time.

`Application Directory`
> This option specifies the directory with the applications the PIPE module can launch. If an empty string is specified for this option, all messages directed to the PIPE module are rejected.

`Processing Time Limit`

This option limits the time an external application uses to process a message. If an external application does not complete within the specified period of time, the application process is interrupted and the message is rejected.

| Submitting |
| --- |
| Check Directory every:    30 sec |

Check Submitted Directory

This option specifies how often the PIPE module should scan the Submitted directory and deliver the .sub files stored there.

# Foreign Queue Processing

In emergency situations you may need to process an additional Queue directory without stopping your server. These situations include a server hardware failure when the rescued Queue files should be processed with some other, already running server.

To process an additional `Queue` directory, move it to the *base directory* of a running server as the `ForeignQueue` directory. If you prefer to use symbolic links, make sure that the `Queue` and `ForeignQueue` directories are created on the same file system.

Every 3 minutes the PIPE module checks if the `ForeignQueue` directory exists in the server *base directory*. If the `ForeignQueue` directory is found, the module moves all `.msg` files from the `ForeignQueue` to the `Queue` directory (it can rename the files in the process), and deletes all `.tmp` files found in the `ForeignQueue` directory. Files with other extensions are left in the `ForeignQueue` directory.

All moved `.msg` files are submitted to the Server kernel, into the ENQUEUER queue, and the Server starts to process them in the same way it processes all submitted messages.

When the scanning process is completed, the `ForeignQueue` directory can be removed from the *base directory*.

CommuniGate **Pro**

# Real-Time Signals

- **AORs**
- **Signals**
- **Processing Requests**
- **Automated Processing (Rules)**
- **Forking**
- **Configuring the Signal Component**
- **Sending Signals from Accounts**
- **Sending Signals to Accounts**
- **Sending Signals to Remote Systems**
- **Service Calls**
- **Registrar Services**
- **Event Packages**
  - **Presence**
  - **Message Summary**
  - **Registration**
  - **Dialog**
- **Calls**
  - **Account Call Logs**
  - **Call Detail Records (CDRs)**
- **Local Nodes**
- **Call Legs**
- **Signal API**

One of the main functions of the CommuniGate Pro Server is Real-time communication. Acting as a Signaling Center, the Server receives Real-Time Signals (Requests) from various sources (the SIP and XMPP Modules, sessions, "call legs", internal kernel components, etc.)
The CommuniGate Pro Server either processes (terminates) these Requests itself, or it delivers them to remote entities (via the SIP or XMPP protocol), or it delivers them to internal sessions and "call legs".

For each Requests, the Signal module optionally generates some *provisioning* Responses (such as "Trying" or "Ringing"), and one *final* Response.

## AORs

Users configure their devices (IP phones, AV conferencing tools, Instant Messaging tools) to connect to a selected Server when they go on-line.

Each user has a unique "XMPP identifier" or "SIP identifier", also called AOR (Address of Record).

For session-less protocols such as SIP, the Server registers the users by remembering their "SIP identifier" and the

network (IP) addresses they use.

Each user may have several *registrations* active if that user has several communication devices in the on-line mode (an office IP Phone, a desktop computer, an instant messaging program on a laptop, etc.)

Registrations allow SIP users to communicate with each other without the knowledge of the network addresses being used, using just the "SIP identifiers" (AORs).

AORs have the same form as E-mail addresses: `username@domainName`. The CommuniGate Pro user AOR is the full name of the user Account, so the user SIP AOR name is exactly the same as the user E-mail address.

CommuniGate Pro uses the Router component for all Real-Time Communication operations, so Aliases, Forwarders, and other Routing methods work for Real-Time Communications, too.

---

# Signals

A Signal is a basic Real-time Task. One Real-time entity can send Signals to some other Real-time entity to start, modify, or stop a communication dialog, to deliver a status update, etc.

The sending entity composes a Request object and sends it to the CommuniGate Pro Signal module. The Signal module processes the Request, optionally sends Requests to other entities, and returns a Response object to the sending entity.

Many CommuniGate Pro modules and components can use Signals:

- The SIP Module server subcomponent receives Requests from external Real-time entities (SIP clients, other SIP Servers) using the SIP protocol. When the Signal Module generates a Response object, the SIP Module sends the response back to the external entity.
- The XMPP Module and XIMSS Module send and receive Signal Requests and Responses, and transfer them between the client application and the CommuniGate Pro Server.
- Internal Real-time Node, such as Real-Time Application nodes (such as PBX applications) can send various Signal requests.
- Automated Processing Rules can use Signals (to send Instant Messages as notifications).

---

# Processing Requests

When the Signal Module receives a Request, it calculates the target URI for it. It takes the Request URI (or the first Route URI in the Request Route set), and uses the Router component to detect the Request destination. There are several possible outcomes of this process:

- The Router returned an error code (for example, the Request URI addressed a local Account that does not exist). This error code is returned to the Signal source, and processing stops.
- The Router returned a non-local address (an address in a non-local Domain). The Request is passed to the SIP Module for relaying. The Response returned with the SIP module is passed to the Signal source.
- The Router returned an address of a local internal Real-time Node. The Request is passed to that Node for processing. The Response returned with the Node is passed to the Signal source.
- The Router returned a local Account address, and the Request can be processed with the Signal Module itself. The Request is processed, and the Response is returned to the Signal source. This case includes REGISTER-type Requests and other Requests which URIs target local Domains.
- The Router returned a local address, and the Request cannot be processed with the Signal Module itself. The Forking process starts with the Request URI as the initial AOR set.

- The Router returned the `null` address. The OK code is returned to the Signal source without further processing.

The following diagram illustrates the Signal flow inside the CommuniGate Pro Server:
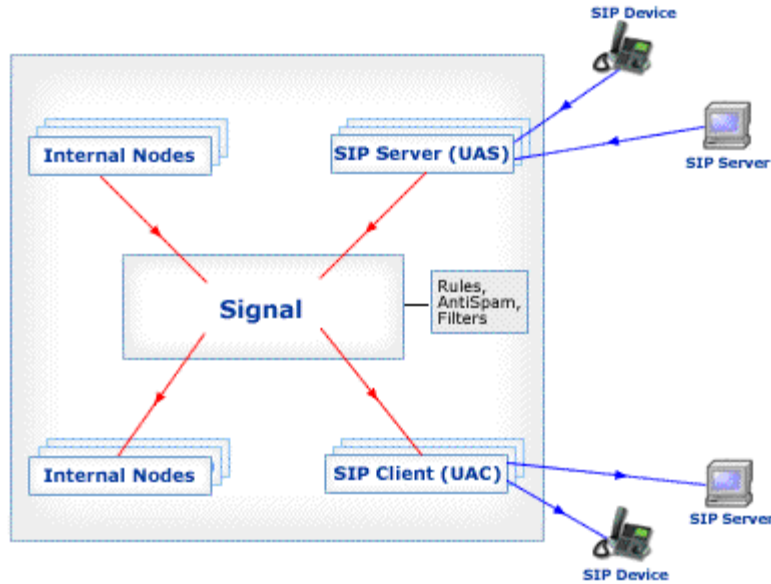


# Automated Processing (Rules)

After an address has been processed with the Router, but before it was relayed to a local or a remote entity, Server-wide and Cluster-wide Automated Signal Processing Rules are applied.

The Rules control how the Request is processed: they can direct it to a different destination, or they can reject the Request.

Only the Dialog-initiating Requests are processed with the Automated Rules.

# Forking

The Signal module maintains the AOR (Address-of-Record) and Contact sets for each Request it processed. The module starts the Forking process by processing addresses in the AOR set.

When a 2xx response is received while processing any AOR, AOR processing stops. If the Request was an INVITE request, all outstanding Requests relayed to other AORs are canceled.

When all AORs have been processed, the module returns the "best" processing result to the Request source.

When an AOR to be processed is Routed to a non-local address, that address is placed into the Contact set.

When an AOR to be processed is Routed to a local Group, all Group members are added to the AOR set.

When an AOR to be processed is Routed to a local Account, all Account active *Registrations* (registered addresses of the Account user devices) are added to the Contact set.

If an AOR already exists in the AOR set, it is not added to the AOR set.

If an address already exists in the Contact set, it is not added to the Contact set.

The Signal module checks each address it adds to the Contact set.
If the new Contact address is a [Local Node](#) address, the Request is passed to that Node for processing.
If the new Contact address is an external address, the Request is passed to the SIP Module for relaying.

When a local or an external entity returns a redirect-type Response, the module checks the initial AOR (the Request URI).

- If the initial AOR was Routed to a local Account or Group, the addresses specified in the redirect-type Response are added to the AOR set and the Forking process continues.
- If the initial AOR was Routed to a remote address, the redirect-type Response is returned to the Request source.

---

# Configuring the Signal Component

You can use the WebAdmin Interface to configure the Signal component. Open the Real-Time pages in the Settings realm, then open the Signals page:

| Processing | | | |
|---|---|---|---|
| Log Level: Failures | Processors: 3 | Object Limit: 1000 | Event Limit: 300 |

Log

> Use this setting to specify the type of information the Signal component should put in the Server Log. Usually you should use the `Failure` (unrecoverable problems only), `Major` (Signal progress reports), or `Problems` (failures, progress reports, and non-fatal errors) levels.
>
> When you experience problems with the Signal component, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case the signal processing internals will be recorded in the System Log. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.
>
> The Signal component records in the System Log are marked with the `SIGNAL` tag.

Processors

> The Signal component uses several simultaneous processors (threads) to process Signal "tasks". One processor can handle several Signal tasks. If you use many time-consuming [Automated Rules](#), you should allow the component to use more processors for signal processing.

Object Limit

> This setting specifies how many Signal "tasks" the component can handle at any given time.
>
> If this limit is exceeded, new Signals are rejected, and an error is returned to the Signal sender.

Event Limit

> This setting specifies the critical number of unprocessed events sent to all active Signal "tasks".
>
> If this number is exceeded, the component is overloaded (the component Processors cannot handle events as they appear), new Signals are rejected, and an error is returned to the event sender.

Use the Signal Task panel to configure Signal processing by Signal Tasks.

| Signal Task | | |
|---|---|---|
| AOR Limit: 30 | Time Limit: 15 min | |
| Active Fork Limit: 30 | | Authenticate all outgoing calls |

AOR Limit

> This setting limits the number of AORs that can be processed for a single operation. It limits the total number of forks, redirects, etc. for a single request.

**Active Fork Limit**
      This setting limits the number of concurrent forks for a single request.

**Time Limit**
      This setting specifies the default time limit for request processing.

**Authenticate all outgoing calls**
      The CommuniGate Pro Server requires user authentication for certain Requests:

* when requests are sent remotely via SIP, authentication is performed with the [SIP module](#) server component
* the [XIMSS](#) and [XMPP](#) modules send all Signals authenticated, on behalf of the logged-in Account
* the [Real-Time Applications](#) send Signals authenticated, on behalf of their *current Account* (unless the Application has impersonated as "nobody")

This option requires authentication for all calls (initial INVITE requests) send from this Server Account addresses.

When a call-type Signal (an initial INVITE request with audio-video Session Descriptor) is authenticated, the Signal Module performs additional processing of the authenticated Account.

The Signal component can limit the number of "calls" an Account can place over a specified period of time. If more calls are placed, they are rejected without processing.
This limit can be set individually, for each Account, or server/cluster-wide as well as Domain-wide default settings can be used. See the [Accounts](#) section for more details.

---

# Sending Signals to Accounts

If the first AOR in the set (the AOR specified in the Request URI) is a local Account address, and the Request is an INVITE request, the Account device registration is retrieved, along with certain Account settings and preferences.

An Account can have [Automated Signal Rules,](#) which are retrieved with Account device registrations. These Rules are "mixed" with the Domain-wide Rules specified for the Account Domain and are applied to all requests sent to the Account.

The Signal component controls how many "calls" (initial INVITE requests with audio-video Session Descriptors) an Account receives over a specified period of time. If more calls are received, they are rejected without processing. This limit can be set individually, for each Account, or server/cluster-wide as well as Domain-wide default settings can be used. See the [Accounts](#) section for more details.

---

# Sending Signals to Remote Systems

If a Signal is to be relayed to a remote system, the Signal component directs it to some signaling module.

The Signal target address can be explicitly routed to some module. For example, all domains with the `._xmpp` suffix are routed to the [XMPP](#) module.

If the Signal target address is not explicitly routed to some module, the Signal component check the target domain name

* If the domain name is an IP address, the Signal is routed to the [SIP](#) module.

- If the domain name has SIP SRV records in the DNS, the Signal is routed to the SIP module.
- If the domain name has XMPP SRV records in the DNS, the Signal is routed to the XMPP module.
- In all other cases the Signal is routed to the SIP module.

# Service Calls

If a Request is routed to a *nnnn name in any of the local Domains (where *nnnn* is any sequence starting with a decimal digit), the Request is rerouted to its originator (the Request From: address).

This feature allows users to dial a *nnnn number to request a service from the Server. The Request is routed to the user's own Account, where it starts the Self-Call application. The application provides the requested service using the Request To: address to learn the dialed "service option" number.

# Registrar Services

Communication devices used by CommuniGate Pro users should register themselves before they can receive Requests from other entities.

Registration Requests require user authentication.

One Account credentials can be used to modify registrations for a different Account, if the authenticated Account has the CanImpersonate access right for the target Account Domain.

To configure the Registrar Service options, open the Real-Time pages in the WebAdmin Settings realm, and select the Signals page.

| Services | | | | | | |
|---|---|---|---|---|---|---|
| Registration Time: | Minimal: | 30 sec | Default: | 30 min | Maximal: | 24 hour(s) |
| | Random delta: | 0 sec | | | | |
| MS RTC: | Integrate with Roster: | Enabled | | | | |

Registration: Minimal
 Use this option to specify the minimal Registration expiration time period accepted.
 If a Registration Request contains a shorter expiration time, the Request is rejected, and the Response sent specifies this minimal accepted time. The entity (usually - a SIP device) can resend the Registration Request with an adjusted expiration time.

Registration: Default
 This option value is used when a Registration Request does not specify a Registration expiration time.

Registration: Maximal
 Use this option to specify the maximal Registration expiration time period accepted.
 If a Registration Request contains a longer expiration time period, the period is shorten to the specified value.

Registration: Random delta
 If this option is set to a non-zero value X, the requested Registration expiration time period is decreased by a random value, from 0 to X.
 Large sites can use this feature to soften "registration storms", when a large number of devices starts to register at the same time (after a network failure or a maintenance window).
 The requested expiration time period is decreased only if it is larger than 2*X and larger than Minimal+X.

**Note:** the Registrar component supports the SIP `gruu` (Globally Routable User Agent (UA) URIs) extensions.

**Note:** the Clients that use for registration domain names starting with `srtp.` or `dtls.` prefixes are marked as ones expecting secure media offers in incoming calls. Alternatively, if [Account Detail](#) is enabled on the server and authentication name during registration contained as detail string `srtp` or `dtls`, then the Client will also be offered secure media on incoming calls.

# Event Packages

The Signal Module supports several Event Packages. When it receives a SUBSCRIBE Request targeting a local Account, it may process the Request itself, without forwarding it to the Account registered entities.
**Note:** as a workaround for many buggy clients, the Module accepts SUBSCRIBE Requests sent to local Domains (instead of local Accounts). For these requests, the address in the `To` field is used to replace the domain-type address in the Request URI.

The Signal module maintains "tuple" states for each Account, for each Event Package it supports. The module allows one or several entities to update the "tuples", and it aggregates them into one Account "state information" for the Package. When the aggregated "state information" changes, the module sends NOTIFY requests with the updated state to all subscribed entities.

The Signal module allows external entities to modify "tuples" using PUBLISH requests.

The Signal module allows external entities to modify "tuples" for certain Packages using non-standard SERVICE requests.

The Signal module provides "watcher" packages for all Event Packages. These packages provide the information about current subscribers to the "base" Event packages. "Watcher of watcher" packages are implemented, too.

Unless explicitly specified otherwise, subscription to a Account Event package is available only to the Account owner, to other Accounts with the `CanImpersonate` [Access Right](#) for the Account Domain, and to other Accounts that have the Delegate [Account Access Right](#) for this Account.
The same restrictions are applied to subscriptions to all "watcher" packages.

## Presence

The Signal Module implements a Presence Server. The Module supports the `PIDF`, `CPIM-PIDF` and `XPIDF` Presence Information formats.

The Signal Module provides special processing for the REGISTER requests. If an external entity (a SIP device) indicates support for the SUBSCRIBE method, the module establishes a presence subscription dialog with that entity. The module then processes all NOTIFY requests sent by that entity to maintain its Presence "tuple" or "segment".

A Presence segment value is a string from a fixed set, listed here starting from the lowest priority value to the highest priority one:

- `offline`
- `away`
- `out-lunch`
- `in-meeting`
- `be-back`
- `online`
- `on-phone`

- `busy`

The event aggregated value is a segment value with the highest priority, or `offline` if no segment exists.

When a NOTIFY request is composed, the aggregated value is converted into a presence document in one of the supported formats.

Subscription to the Event package is available to anyone, subject to the standard subscription approval process.

## Message Summary

The Signal Module implements the MWI (Message Waiting Indication) Service. The Module supports the `simple-message-summary` Information format.

The Server itself maintains the "INBOX" tuple/segment for this Event package. The segment value is set to an array:

- the first element is the number of INBOX messages that have the $Media flag set and do not have the Seen flag set (the number of unread media messages);
- the second element is the number of INBOX messages that have the $Media flag set (the total number of media messages);

The event aggregated value is an array containing the by-element sum of all segment array values.

When a NOTIFY request using the `simple-message-summary` Information format is composed, the first aggregated array element value is used as the number of new voice messages, and the difference between the second and the first elements is used as the number of old voice messages.
If the first array element value is not zero, the `Messages-Waiting` element is set to `yes`, otherwise it is reset to `no`.

## Registration

The Signal Module implements the Registrar Monitoring Service. The Module supports the `reginfo+xml` Information format, and it can inform network entities (such as SIP clients) about all other entities registered with an Account.

## Dialog

The Signal Module implements the Dialog Monitoring Service. The Module supports the `dialog-info+xml` Information format, and it can inform network entities (such as SIP clients) about all dialogs active with an Account.

Subscription to the Dialog package is available to the Account owner, to other Accounts with the `CanControlCalls` Domain Access Right, and to other Accounts with the CallControl Access Right granted by with Account owner.

---

# Calls

The CommuniGate Pro Server creates Call objects when processing call-starting INVITE requests. Call objects contain information about the calls initiated with these requests.
Call objects are destroyed when these calls are terminated.

If the caller is authenticated, the call status is recorded in the caller Account data, and a CDR record is generated. When a call is completed, an accounting record for this outgoing call is added to the caller call log.

If the call target (callee) is authenticated, the call status is recorded in the callee Account data, and a CDR record is generated. When a call is completed, an accounting record for this incoming call is added to the callee call log.

When the authenticated peer (the caller or the callee) places a call on hold, their Account "Hold Music" preference is used.

If it is not set to `disabled`, the specified sound file is read from the Account File Storage. If the file name is "*" or if the specified file is not found, the `HoldMusic` file is read from the Account Domain [Real-Time Application Environment](#).

When the sound file is read, a [Media Channel](#) is created and instructed to play the retrieved file to the other peer.

You can use the WebAdmin Interface to configure the Call Manager component. Open the Real-Time pages in the Settings realm, then open the Signals page:

| Calls | | | |
|---|---|---|---|
| Log Level: | Low Level | Time Limit: | 24 hour(s) |
| Signal Idle Timeout: | 60 min | Media Idle Timeout: | 15 min |

`Log`

Use this setting to specify the type of information the Call Manager component should put in the Server Log. Usually you should use the `Failure` (unrecoverable failures only), `Major` (failures and major events), or `Problems` (failures, major events, and non-fatal errors) levels.

The Call Manager component records in the System Log are marked with the `DIALOG` tag.

`Time Limit`

This setting specifies a time limit for Calls. When this limit is exceeded, the Call object is removed. Keep this limit high enough to avoid call interruptions.

`Signal Idle Timeout`, `Media Idle Timeout`

If there is no signal sent within a Call for the specified period of time, and, if a Media Proxy is associated with that Call, there no media data to relay for the specified period of time, a call termination (BYE) request is sent to both sides of the call, and the Call object is removed. Keep these limits high enough to avoid call interruptions when a call is placed on hold.

## Account Call Logs

The Account Call Logs are stored as text files in the Account [File Storage](#).

These files have the following format:

> 2_dd-mmm hh:mm:ss_direction_peer_callId_callTime_alertTime_errorCode[_programName]

where:

`_`

is the tabulation symbol (symbol code is 0x09)

`2`

record format version

`dd`

2-digit month day number

`mmm`

3-letter month name

`hh`, `mm`, `ss`

2-digit numbers for the call termination hour (00..23), minute (00..59), second (00..59)

`direction`

1-letter call direction: `I` - incoming, `O` - outgoing

`peer`

E-mail address of the call peer:

> <username@domainName >

or

"*real name*" <*username@domainName*>

callId

call `Call-ID` string.

callTime

call duration (in seconds). The time between the moment the call connected and the moment the call disconnected. If this call has not succeeded, this field is empty.

alertTime

call alerting time (in seconds). The time between the moment the call started and the moment the call connected or (if the call has not succeeded) the moment the call failed.

errorCode

a string with the error code for the call attempt failure or the call disconnect reason. If the call has completed without an error, this field is empty.

programName

this optional field contains the name of the [Real-Time Application](#) that handled this call.

# Call Detail Records (CDRs)

The Signal module can generate Call Detail Records for INVITE and BYE transactions it processes. It can also generate Call Detail Records for completed calls using the information in the [Call](#) objects.

CDRs can be generated and stored in special [supplementary Log](#) files.

| Monitoring | |
|---|---|
| Record Call CDRs | Record INVITE/BYE CDRs |

Record Call CDRs

Select this option to generate CDR Log files for Calls (Call objects).

Record INVITE/BYE CDRs

Select this option to generate CDR Log files for INVITE and BYE requests.

When the [External CDR Processor](#) Helper application is enabled, the Signal Module generates all types of CDRs and sends them to that application.

CDR data text for INVITE/BYE requests has the following format (the tabulation symbol is used as the field separator):

**01** *NNN-method dialog-identifier parties network-addresses flags*

where:

01

the CDR format version number

*NNN*

the transaction result code (numeric)

*Method*

the transaction operation/method (INVITE, BYE).

*dialog-identifier*

the transaction `Call-ID` field, the `From` field `tag` parameter, the `To` field `tag` parameter, enclosed in the < and > symbols.

Note: If a call is terminated by the callee, the *fromTag* of the BYE transaction is the *toTag* of the INVITE transaction and vice versa.

*parties*

the transaction `From` and `To` field URIs addresses, enclosed in the < and > symbols.

*network-addresses*

the network (IP) address the transaction request and responses were received from, enclosed in the **[** and **]** symbols.

*flags*

optional fields separated with tabulation symbols:

`[auth:`*`accountName`*`]`
    this element is added if the transaction request is authenticated. The *accountName* is the name of the authenticated CommuniGate Pro Account.
`[redir:`*`accountName`*`]`
    this element is added if the transaction request was redirected from a local Account. The *accountName* is the name of that CommuniGate Pro Account.
`[billing:`*`billingData`*`]`
    this element is added if the transaction request has a P-Billing-Id field. The *billingData* string is the field content.
`[referred:`*`referredby`*`]`
    this element is added if the transaction request has a Referred-By field. The *referredby* string is the field content.

CDR data text for calls has the following format (the tabulation symbol is used as the field separator):

**02** *callType accountName alertTime connectTime reason dialog-identifier* [**"**_fromName_**"**] **<**_from_**>** [**"**_toName_**"**] **<**_to_**>**[**[**_toProgramName_**]**] *flags*

where:

`02`
    the CDR format version number
*callType*
    the call type (INP, OUT).
*accountName*
    the full Account name (*accountName@domainName*).
*alertTime*
    the time period (in seconds) between the time the call was initiated and the time the call was accepted or rejected.
*connectTime*
    the time period (in seconds) between the time the call was accepted and the time the call was terminated. If the call was not accepted, this field contains the – (minus) symbol.
*reason*
    the reason the call was rejected or terminated.
*dialog-identifier*
    the call `Call-ID` field, the `From` field `tag` parameter, the `To` field `tag` parameter, enclosed in the **<** and **>** symbols.
*from*, *to*
    the transaction `From` and `To` field URIs.
*fromName*, *toName*
    the transaction `From` and `To` field "real names". These fields are optional. If present, these field values are enclosed into the quotation mark (**"**)symbols.
*toProgramName*
    optional - the name of the PBX Application that has accepted the call.
*flags*
    the same as used with version 01 records.

---

# Local Nodes

The CommuniGate Pro Server dynamically creates, runs, and removes Real-Time Nodes.

A Node is an internal CommuniGate Pro Server object that can receive Signal Requests and produce Responses for those Requests. A Node can also send Requests and process Responses.

Various CommuniGate Pro components and modules use Nodes to implement Signaling functions:

- The [Real-Time Application](#) component creates a Node for each "call leg".
- The [XIMSS](#) session manager creates a Node for each XIMSS "call leg".
- The [Presence](#) subcomponent creates a Node and uses it to manage Presence Event Subscriptions for those devices that cannot proactively update their Presence state.

You can use the WebAdmin Interface to configure the Nodes component. Open the Real-Time pages in the Settings realm, then open the Nodes page:

| Processing | | | |
|---|---|---|---|
| Log Level: Failures | Processors: 3 | Object Limit: 1000 | Event Limit: 300 |

Log
> Use this setting to specify the type of information the Local Nodes component should put in the Server Log. Usually you should use the `Failure` (unrecoverable failures only), `Major` (failures and major events), or `Problems` (failures, major events, and non-fatal errors) levels. When you experience problems with the Local Nodes component, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case the Node processing internals will be recorded in the System Log. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.
> The Local Nodes component records in the System Log are marked with the `NODE` tag.

Processors
> The Local Nodes component uses several simultaneous processors (threads) to process Nodes. One processor can handle several Nodes tasks.
> If you use many Nodes implementing time-consuming operations (complex Real-Time Applications, etc.) you should allow the component to use more processors.

Object Limit
> This setting specifies how many Nodes the component can handle at the same time. If this limit is exceeded, attempts to create new Nodes result in an error.

Event Limit
> This setting specifies the critical number of unprocessed events (sent to all active Nodes). If this number is exceeded, the component is overloaded (the component Processors cannot handle events as they appear), and attempts to create new Nodes result in an error.

# Call Legs

The Server creates special Local Nodes called *Call Legs*. Each Call Leg terminates signaling for one phone call. Each [PBX Task](#) is a Call Leg, and each [XIMSS](#) session can create one or more Call Legs to handle phone calls initiated or accepted by the XIMSS session user.

The settings in the Call Legs panel are applied to all types of Call Legs:

| Call Legs | | | |
|---|---|---|---|
| Session Expiration: | Minimal: 90 sec | | Default: 3 min |
| Hop Limit: | 20 | | |

Session-Expiration
    Use this setting to specify how often the parties should exchange the INVITE (or OPTION) requests to verify
    that the call has not been broken.

Hop Limit
    Use this setting to specify the `Max-Forwards` field value for the requests sent with Call Legs.

# Signal API

The Signal tasks can receive "events" from other CommuniGate Pro components, such as [CG/PL](#) tasks and [XIMSS](#)
client sessions. To send an event, the sender should obtain a *signal handle*.

A Signal task accepts the following events:

decline
    This event causes the Signal task to cancel Signal Request processing and to reject it with the 603 Declined error
    code.
fork
    This event data should be a string or an array of strings. The Signal Request is forked to the URIs specified with
    these strings.
redirect
    Same as `fork`, but all currently "forked" Requests are canceled.

CommuniGate **Pro**

# Automated Signal Processing Rules

- **Applying Signal Rules**
- **Specifying Signal Rules**
- **Rule Conditions**
- **Rule Actions**
- **Macro Substitution**
- **Logging Rules Activity**

The CommuniGate Pro Server can automatically process Signals using sets of Automated Rules. Rules are not applied to Signal Requests sent inside already established Dialogs.

The system-wide (Server-Wide and Cluster-wide) Rules are applied to all Signals sent to the Server and/or to the Cluster.

When a Signal request is directed to a CommuniGate Pro Server Account, the Account-level Rules are applied.
The Account-level Rules are the Rules specified for the particular Account along with the Rules specified for the Account Domain.

# Applying Signal Rules

Each signal Rules has its *stage* and *priority*. The stage specifies when the Rule should be applied:

- immediately when the Signal is received
- after the Signal is being processed for some time
- when the Signal fails with the No Answer, Busy, or some other error condition.

Within each stage the Rules are applied according to their priority.

When Server, Cluster, Domain, and Account Rules are "merged", they are grouped based on the stage value. Within each stage, the Rules are applied in the following order:

- the Server and Cluster Rules with the priority > 5
- the Domain Rules with the priority > 5
- all Account Rules
- the Domain Rules with the priority <= 5
- the Server and Cluster Rules with the priority <= 5

**Note:** when Signal processing starts, all Server and Cluster Rules for the "immediate" stage are applied first, and only then processing continues. The Signal can be directed to a local Account, and the Account and Account Domain Rules are "merged" with the Server and Cluster Rules. But at that moment all Server and Cluster Rules for the "immediate" stage have been already applied and removed from the "merged" set.

# Specifying Signal Rules

System administrators can specify Server-wide and Cluster-wide Signal Rules. Use the WebAdmin Interface to open the Real-Time pages in the Settings realm, and click the Rules link.

System and Domain Administrators can specify Account Rules using links on the Account Settings WebAdmin Interface pages.

Account users can specify their Rules themselves, using the WebUser Interface. System or Domain Administrators can limit the set of Rule actions a user is allowed to specify.

System and Domain Administrators can specify Domain-Wide Rules using the Rules links on the Domain Settings pages.

See the generic Automated Rules section to learn how to set Rules.

# Rule Conditions

Each Rule can use universal conditions specified in the Rules section.
Additionally, the following Rule conditions can be used in Signal processing Rules:

`Operation [is | is not | in | not in]` *string*
     This condition checks if the Request method is or is not equal to the specified *string*.

     Sample:

| Operation | is |
|-----------|----|

     This condition will be met for all INVITE Requests.

`CallType [is | is not | in | not in]` *string*
     This condition checks if the Request SDP type is (or is not) equal to the specified *string*.
     The SDP type is `AV` if the request method is INVITE and it contains at least one audio or video media channel;
     The SDP type is `IM` if the request method is MESSAGE or if the request method is INVITE and it contains an IM channel.
     The SDP type is an empty string in all other cases.

     Sample:

| CallType | is |
|----------|----|

     This condition will be met for all audio/video INVITE Requests.

`Target Address [is | is not | in | not in]` *string*

This condition checks if the Request URI is (or is not) equal to the specified *string*.

Sample:

| Target Address | is |
|---|---|

This condition will be met for all signals coming to on any vm.communigate.com account.

`From [is | is not | in | not in]` *string*

This condition checks if the Request `From` address is (or is not) equal to the specified *string*.

Sample:

| From | is |
|---|---|

This condition will be met for all signals coming from any account of any of communigate.com subdomains.

`To [is | is not | in | not in]` *string*

The same as above, but the Request To field address is checked.

`'From' Name [is | is not | in | not in]` *string*

The same as above, but the instead of the address, the "address comment" (the real name) included in the From address is checked.

Sample:

| 'From' Name | is |
|---|---|

This condition will be met for Requests with the following `From:` addresses:

```
From: <sip:jsmith@company.com> (John J. Smith)
From: "Bill J. Smith" <sip:b.smith@othercompany.com>
From: Susan J. Smith <sips:susan@thirdcompany.com>
```

`Authenticated [is | is not | in | not in]` *string*

This condition checks the Request authentication. If the Request has been authenticated by this CommuniGate Pro Server, the authenticated Account name (`account@domain`) is compared with the specified *string*.

Sample:

| Authenticated | is |
|---|---|

This condition will be met for all signals coming from any account on any of communigate.com subdomains.

`Presence [is | is not | in | not in]` *string*

This condition checks the aggregated presence of the request target. This condition can be used in the Domain- and Account- level Rules only.
The Presence is one of the strings `online`, `busy`, `away`. The Presence is an empty string when the call target is offline.

Sample:

<div style="border:1px solid #888; padding:10px">
Presence        in
</div>

`Active Devices [is | is not | less than | greater than | in | not in]` *number-string*
This condition checks the total number of Registered Devices for the request target. This condition can be used in the Domain- and Account- level Rules only.
When a call is forked to a different destination, the total number of Registered Device is increased by the number of registered devices for the new request target.
When a call is redirect to a different destination, the total number of Registered Device is reset to zero, and then it is increased by the number of registered devices for the new request target.

`Device Type [is | is not | in | not in]` *string*
This condition checks the type of device sending the request (the User-Agent request field).

`Request Field [is | is not | in | not in]` *string*
The *string* must contain the Request field name and the field data picture, separated with the colon (`:`) symbol. This condition compares the specified request field data with the data picture. Only simple, unstructured request fields can be compared.

Sample:

<div style="border:1px solid #888; padding:10px">
Request Field        is
</div>

---

# Rule Actions

Each Rule can have zero, one, or several actions. If a Request meets all the Rule conditions, the Rule actions are performed.

You can use all universal actions described in the [Rules](#) section. This section describes the Rule actions that can be used in Signal Rules:

`Stop Processing`
This action should be the last one in a Rule. Execution of this Rule stops and no other (lower-priority) Rules are checked for this Signal at this time. Signal processing proceeds.

`Discard Rules`
This action should be the last one in a Rule. Execution of this Rule stops and no other (lower-priority) Rules are checked for this Signal, and all remaining Rules (scheduled to apply at later times or if the Signal fails) are removed, too. Signal processing proceeds.

`Redirect to` *addresses*
The Signal is redirected to one or several specified addresses: the current Signal "destination set" is cleared, and the specified addresses form the new "destination set".
Each address should be specified as a `sip:`, `sips:`, or `tel:` URI. If several addresses are specified, they should be separated with the comma (`,`) sign.

Sample:

| | |
|---|---|
| Operation | is |
| Authenticated | is not |

| |
|---|
| Redirect to |
| |

This Rule will direct all INVITE Signals to the adManager@domain.com address, unless the Signal authenticated source is adManager@domain.com.

You can use this Rule to implement an advertising server: the adManager@domain.com Account can start a PBX application that will play some media to the caller, and then, acting as a B2BUA, will connect the caller with the original destination. The calls made by that application will have adManager@domain.com as their authenticated source, so this Rule will not redirect them.

```
Fork to addresses
```
Same as `Redirect to`, but the specified addresses are added to the current "destination set" without clearing it first.

```
ParlayDirect parameters
ParlayNotify parameters
```
These actions implement the "CallDirection" and "CallNotification" [ParlayX](#) Interfaces. The parameters setings specifies the request operation to use, the Parlay "correlator", and the URL to send the request to.

# Macro Substitution

Parameter strings for some actions can include "macro" - symbol combinations that are substituted with actual data before the parameter is used with the Rule action.

The following symbol combinations are available:

`^F` is substituted with the request From address (including the "Real name" part, decoded and converted to UTF-8)
`^E` is substituted with the request From address (the E-mail address only)
`^t` is substituted with the RFC822-formatted current-time.
`^I` is substituted with the request Call-ID.
`^R` is substituted with the request To address (the E-mail address only)
`^M` is substituted with the request Method.
`^U` is substituted with the request URL.
`^^` is substituted with a single ^ symbol.

# Logging Rules Activity

The [Signal](#) component records system-wide Rules activity in the Log. Set the Signal Log Level to `Low-Level` or `All Info` to see the Rules checked and the actions executed.

Overview Rules **IM & Presence** SIP XMPP SMPP PSTN NAT Media Parlay

# Instant Messaging and Presence

- Roster
- Presence
- Instant Messages
- Preferences

The CommuniGate Pro Real-Time component can receive, send, and relay Instant Message requests. These requests usually contain small text strings, but they also can contain service information (such as "user is typing a message"), and/or some non-text data.

The CommuniGate Pro Real-Time component maintains and distributes the "presence" information, and exchange the presence updates with external systems.

# Roster

Each CommuniGate Pro Account has a Roster - a set of "Buddies" - `username@domainname` addresses with the associated status information.

When the Account user adds a Buddy to the Roster (using any XIMSS, XMPP, or SIP client, or the WebUser Interface), a Signal request is sent to the Buddy address. If the receiver confirms this request "to become friends", then the Buddy status in the Roster status information becomes "confirmed", and both sides can see the Presence status of each other.

Roster Buddies can be assigned to one or several Roster Groups, and client applications usually show Roster Buddies sorted by Groups.

# Presence

Each CommuniGate Pro Account can have zero, one, or several client application connected to it using SIP, XMPP, XIMSS or other real-time protocols. Each client can specify its "presence state" - such as "online", "away", "busy", etc.

The CommuniGate Pro Server "aggregates" all reported "presence states" to compose the presence state of the Account itself. For example, if at least one client application reports the "busy" state, the Account state is set to "busy", otherwise if there is at least one client application reporting the "online" state, the Account state is set to "online", etc. If there is no client application connected to (or registered with) the Account - the Account state is set to "offline".

When presence information is distributed, the CommuniGate Pro server adds the "hash" of the Account avatar (read from the Account File Storage), so avatar changes can be detected by other users.

# Instant Messages

When an Instant Message request is delivered to a CommuniGate Pro Account, it is processed as any other Signal Request - Signal Automated Rules are applied, the request is forked to all active XIMSS and XMPP sessions, and, optionally, to registered SIP devices.

Incoming and outgoing IMs are stored in the Account File Storage.

If an IM cannot be delivered because there is no session or device to relay it to, the message is appended to a special file in the Account File Storage.
When the Account user launches an XMPP or XIMSS client application, all stored IMs from that file are delivered to the user via that client application, and the file is removed.

Incoming IMs can be rejected without processing if the IM sender is not a confirmed Buddy in the Account Roster. See the Preferences section below.

# Preferences

The Account user can control Instant Message processing using the following Preferences:

| Instant Messaging | | |
|---|---|---|
| | Accepted Senders besides Buddies | default(everybody) |
| | IM Offline Storage | default(Enabled) |
| | Always Receive to All Devices | default(No) |

`Accepted Senders besides Buddies`
> This setting controls incoming IMs from addresses not included into the Account Roster. The following values are supported:

> - everybody
>   - all Instant Messages are accepted
> - authenticated
>   - Instant Messages from the same CommuniGate Pro system Accounts are accepted.
> - same domain
>   - Instant Messages from Accounts in the same CommuniGate Pro Domain are accepted.
> - nobody
>   - Instant Messages are rejected

`IM Offline Storage`
> If this option is enabled, and there is no active XMPP or XIMSS session, nor any SIP device registered (if IM delivery to SIP devices is enabled), then the "no address found" error is not returned to the sender. Instead, the IM is stored in the File Storage file, and the positive response is returned to the IM sender.

`Always Receive to All Devices`

If this option is set to Yes, then incoming Instant Messages are always delivered to all active XMPP and XIMSS sessions and registered SIP clients with messaging capability (if delivery to SIP clients is enabled). Otherwise, messages sent within conversation are delivered to the session or the device specified by the sender.

**CommuniGate Pro**

| Introduction | | Overview | Rules | IM & Presence | SIP | XMPP | SMPP | PSTN | NAT | Media | Parlay |
|---|---|---|---|---|---|---|---|---|---|---|---|

Introduction
Installation
SysAdmin
Network
Objects
Storage
E-mail
Real-Time
Access
Services
Directory
Clusters
Applications
Miscellaneous
Licensing

WebMail
Pronto!
PBX

# SIP Module

- **Session Initiation Protocol (SIP)**
- **SIP Transport Settings**
- **SIP Server Settings**
- **SIP Client Settings**
- **Microsoft® Windows Messenger Support**
- **SIP Devices Support and Workarounds**
- **Routing**
- **Monitoring SIP Activity**

The CommuniGate Pro SIP Module implements the SIP Internet protocols via IP networks.

The module is used to receive Signal Requests from remote entities, and to send Signals to remote entities.

The SIP protocol does not include the protocols required for actual data transfer (media transfer protocols). Instead, the SIP protocol allows all participating parties to find each other on the network, to negotiate the media transfer protocol(s) and protocol parameters, to establish interactive real-time sessions, and to manage those sessions: add new parties, close sessions, update session parameters, etc.

# Session Initiation Protocol (SIP)

The CommuniGate Pro SIP Module implements the SIP protocol functionality. The module uses TCP and UDP listeners to receive SIP request and response packets via these network protocols. It also sends the response and request packets via the TCP and UDP network protocols.

The SIP module parses all received SIP packets, and uses the module subcomponents to process the parsed packets. Request packets are submitted to the SIP Server subcomponent, to a new SIP Server transaction or to an existing one.
The SIP Server component uses the Signal Module to process the request. The responses generated with the Signal module are submitted to the SIP Server transaction, and the SIP Server sends them back to the source of the SIP request.

The Signal module can send a Request to a remote SIP device or to a remote SIP server. The module uses the SIP Client subcomponent to create a SIP Client transaction. This transaction is used to send a SIP Request via an Internet protocol, and to process the Responses sent back.

SIP Request packets received with the SIP Module are submitted to the SIP Server subcomponent, while SIP Response packets are submitted to the SIP Client subcomponent, with two exceptions:

- if no Client transaction can be found for a Response packet, the packet is relayed "upstream" by the SIP Module itself, without using the Signal module.
- if no Server transaction can be found for an ACK Request, a SIP Client transaction is created to relay the ACK Request "downstream".

The CommuniGate Pro SIP module supports UDP and TCP communications, and it also supports secure (TLS) communications over the TCP protocol.

The CommuniGate Pro SIP module supports *near-end* and *far-end* NAT traversal, enabling SIP communications for both large corporations with many internal LANs, as well as for home users connecting to the Internet via "dumb" NAT devices.

The session initiation schema described above works correctly only if both parties can communicate directly. If there is a firewall or a *NAT* device between the parties, direct communication is not possible. In this case, the CommuniGate Pro SIP module builds and manages *media proxies*, relaying not only the SIP protocol requests and responses, but the media data, too.

---

# SIP Transport Settings

Use the WebAdmin Interface to configure the SIP module. Open the Real-Time pages in the Settings realm, then open the SIP pages.

Click the Transport link to open the SIP Transport Settings.

## Transport

The Transport panel allows you to configure the network-level options for SIP packet receiving:

| Transport | | |
|---|---|---|
| Log Level: Major & Failures | **UDP Listener** | Dump Call-Related Packets: Disabled |
| Enqueuers: 0 | **TCP Listener** | Channels: 5 |
| Enqueued Limit: 3000 | | Idle Timeout: 10 minutes |
| Use Short Field Names: Disabled | | |
| UDP TOS Tag: OS default | UDP Request Size Limit: | LAN:  WAN: |

Log Level
> Use this setting to specify the type of information about SIP packets and SIP transport the module should put in the Server Log. Usually you should use the Failure (unrecoverable problems only), Major (session establishment reports), or Problems (failures, session establishment and non-fatal errors) levels.
> When you experience problems with the SIP module, you may want to set the Log Level setting to Low-Level or All Info: in this case the packet contents and other details will be recorded in the System Log. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.
> The SIP module transport records in the System Log are marked with the SIPDATA tag.
> Generic SIP information records have the SIP tag.

Dump Call-Related Packets
> When this setting is enabled, all packets related to VoIP calls (INVITE, BYE, etc) are written to the log as if Log Level were set to All Info.

UDP
> To configure the UDP transport, click the UDP listener link. The [UDP Listener](#) page will open. By default, the SIP UDP port is 5060.

TCP
> To configure the TCP transport, click the TCP listener link. The [TCP Listener](#) page will open. There you can specify both secure and clear-text TCP ports. By default, the clear-text SIP TCP port is 5060, and the SIP TLS port is 5061.

Input Channels
> Use this option to specify the maximum number of TCP communication channels the module can open. If the number is exceeded, the module will reject new incoming TCP connections.

Idle Timeout
> Use this option to specify when the SIP module should close a TCP communication channel if there is no activity on that channel. This helps to reduce the resources used for TCP communication channels on large installations. On the other hand, some SIP clients may not function properly if the server closes its TCP connection on a time-out.

Enqueuers
> When this option is set to a non-zero value, received packets are not processed immediately: they are placed into a special queue and the receiving thread becomes ready to receive a new packet immediately. The option specifies a number of additional threads that take packets from that queue and process them, sending them to Server or Client SIP transactions.

Enqueued Limit
> When packets are not processed immediately, but placed into a special queue first (see above), this option limits the size of that queue. When the number of packets in the queue exceeds this limit, new received packets are dropped. You may want to increase the number of Enqueuers in this situation.

Use Short Field Names
> If this option is enabled, all SIP packets (client requests and server responses) the Server generates will use alternative (1-symbol) packet header field names. You may want to enable this option to decrease packet sizes.

UDP Request Size Limit
> Use this option to specify the size for the largest UDP packet that can be sent within your LAN and outside your LAN. If the SIP module needs to deliver a packet and the protocol is not explicitly specified, the SIP module uses the UDP protocol, unless the packet size is larger than the specified limit. In the latter case the TCP protocol is used.

UDP TOS Tag
> Use this setting to specify the TOS tag for all outgoing SIP UDP packets. This tag can be used to set the SIP traffic priority on your LAN.

---

# SIP Server Settings

Use the WebAdmin Interface to configure the SIP module. Open the Real-Time pages in the Settings realm, then open the SIP pages.

Click the Receiving link to open the SIP Server (UAS) Settings.

# Transactions

The Transactions panel allows you to specify how the SIP Module handles SIP server (UAS) transactions.

| Server Transactions | | | |
|---|---|---|---|
| Log Level:   Problems | Processors:   3 | Object Limit:   1000 | Event Limit:   300 |

Log
> Use the Log setting to specify what kind of information the SIP Server subcomponent should put in the Server Log. Usually you should use the `Failure` (unrecoverable problems only), `Major` (session establishment reports), or `Problems` (failures, session establishment and non-fatal errors) levels.
> The SIP Server subcomponent records in the System Log are marked with the SIPS tag.

Processors
> Use this setting to specify the number of threads used to process SIP Server transactions.

Object Limit
> Use this setting to specify the maximum number of concurrent server transactions the SIP Module is allowed to handle. When this number is exceeded, incoming SIP packets with new requests are dropped.

Event Limit
> Use this setting to specify the maximum number of unprocessed events sent to all active SIP server transactions. If this number is exceeded, the SIP Server component is overloaded: no new SIP server transactions can be created, and incoming SIP packets with new requests are dropped.

# Protocol

The SIP Module server component implements [Request Authentication](#) for remote clients. If an internal Server component rejects a Request because it does not contain authentication data, the Module adds special fields to the response it sends, facilitating authentication.

| Protocol | |
|---|---|
| Advertise 'Digest' Authentication | Advertise 'NTLM' Authentication |
| Send '100 Trying' for non-INVITEs | Always Send '100 Trying' for INVITEs |

Advertise Digest AUTH
> Select this option to inform SIP clients that the standard DIGEST authentication method is supported.

Advertise Digest NTLM
> Select this option to inform SIP clients that the non-standard NTLM authentication method is supported.

The user name specified in the authentication data is processed using the [Router](#) component, so Account Aliases and Forwarders, as well as Domain Aliases can be used in authentication names.
The specified [Account](#) and its [Domain](#) must have the SIP Service enabled.

All CommuniGate Pro [Account passwords](#) can be used for SIP authentication.

If the CommuniGate Password option is enabled for the specified Account, the SIP module checks if the Account has the `SIPPassword` setting. If it exists, it is used instead of the standard `Password` setting. This feature allows an Administrator to assign a alternative Account password to be used for the SIP authentication only.

Send '100 Trying' for non-INVITEs
> If this option is enabled and the client resends a request, the SIP module sends the 100 ("Trying") response even in the request method is not `INVITE`.

Always Send '100 Trying' for INVITEs
> If this option is enabled, the SIP module always sends the 100 ("Trying") response before it starts to process an `INVITE` request.

# Protection

The SIP Module server component implements several protection techniques:

- UDP packets and TCP connections from Network IP Addresses included into the [Denied Addresses](#) list are dropped without processing.
- When the number of misformed SIP packets received from some Network IP Address address exceeds the specified frequency limit, that Address is added to the [Temporarily Blocked Addresses](#) list.
- When a SIP request is rejected because of some authentication error, the response is sent after some delay, and the sender Network IP Address is added to the [Temporarily Blocked Addresses](#) list if the number of the authentication errors associated with that Address exceeds the specified frequency limit.
- When a SIP request received from Network IP Addresses included into the [Temporarily Blocked Addresses](#) list, they are dropped without processing.

# SIP Client Settings

Use the WebAdmin Interface to configure the SIP module. Open the Real-Time pages in the Settings realm, then open the SIP pages.

Click the Sending link to open the SIP Client (UAC) Settings.

## Transactions

The Transactions panel allows you to specify how the SIP Module handles SIP client (UAS) transactions.

| Client Transactions | | | |
|---|---|---|---|
| Log Level: Problems | Processors: 3 | Object Limit: 1000 | Event Limit: 300 |

Log

    Use the Log setting to specify what kind of information the SIP Client subcomponent should put in the Server Log. Usually you should use the `Failure` (unrecoverable problems only), `Major` (session establishment reports), or `Problems` (failures, session establishment and non-fatal errors) levels.
    The SIP Client subcomponent records in the System Log are marked with the SIPC tag.

Processors

    Use this setting to specify the number of threads used to process SIP Client transactions.

Object Limit

    Use this setting to specify the maximum number of concurrent client transactions the SIP Module is allowed to handle.

Event Limit

    Use this setting to specify the maximum number of unprocessed events sent to all active SIP client transactions. If this number is exceeded, the SIP Client component is overloaded, and no new SIP client transactions can be created.

## Protocol

| Protocol | | | |
|---|---|---|---|
| Force Dialog Relaying | | Relay to Any IP Address for: | clients |
| Relay via: | | Timer B: | 32 sec |
| Send P-Asserted-Identity | | | |

Force Dialog Relaying

    If this option is disabled, the SIP Module introduces itself only into those SIP dialogs that require its participation (such as those involving NAT/Firewall traversal). If this option is enabled, the SIP module introduces itself into all SIP dialogs opened. This feature can be used for troubleshooting, as all details of dialog transactions are recorded in the Server Log.

Relay to Non-Clients

    If this option is set to `anybody`, the SIP Module acts as an Open Relay: it relays any SIP request to any destination.
    To prevent abuse of your Server, allow relaying for `clients` only or for `nobody`.
    The SIP Module will send Requests if at least one the following conditions is met:

- the destination address is listed as a Client IP address.
- the Request is being relayed to devices registered with some Account on your Server.
- the Request is generated by a Local Node (such as a PBX Task).
- the Request sender is authenticated with your Server.
- the Request is received from a network address listed in as a Client IP address (only if this option is set to `clients`).

    If none of these conditions is met, the request is rejected with the 401 ("Authentication required") error code.

Relay via

    Enable this option if you want to relay all outgoing packets via some external SIP server. Note that this setting is not used for addresses explicitly routed to external hosts using the ._via suffix or other routing methods.

Timer B

    This option controls the value of the "Timer B" (specified in RFC3261). It controls the maximum time the INVITE-type transaction will wait for any first response from the called party.
    While the standard specifies the 32 seconds value, we strongly recommend to lower this value to 5-10 seconds: if the remote party does not provide any answer within that time (not even the 100-Trying response), most likely it is down and there is no need to wait for 32 seconds before reporting this to the call originator.
    Lowering this time allows the SIP Client transaction to try other SRV records (if any exists): if this timer is set to 32 seconds, the calling user is likely to give up before the next SRV record is tried.

487-Wait Timer

    When a SIP transaction is terminated by the client, a CANCEL request is generated and sent. This setting specifies for how long the Module should wait for a 487-response from the client.
    If no response is received, the Module generates the 487-response itself.

```
Send P-Asserted-Identity
```
If this option is enabled, and the request sender is authenticated, a `P-Asserted-Identity` field is added to the SIP request sent. The field contains a SIP URI with the authenticated Account full name (`sip:`*`accountName@domainName`*).

# Microsoft® Windows Products Support

The Microsoft "RTC" products (including Windows Messenger) use the standard SIP protocol for audio and video sessions.
These clients use the proprietary SIP protocol extensions for Instant Messaging, Presence, Whiteboard, Remote Assistance and other services.
CommuniGate Pro implements the extensions required to support these applications.

The Windows Messenger versions prior to 5.0 are not supported.

The CommuniGate Pro SIP module should have the Advertise NTLM option enabled.

The Windows Messenger audio and video sessions use standard RTP media protocols and these sessions can be used over a NAT/Firewall.
The Windows Messenger Instant Messaging uses the SIP protocol for media transfer and Instant Messaging sessions can be used over a NAT/Firewall.
The Windows Messenger Whiteboard, Application Sharing, and Remote Assistance sessions use T.120 and non-standard protocols and these sessions can be used over a NAT/Firewall.
The Windows Messenger File Transfer sessions use a non-standard protocol and these sessions currently cannot be used over a NAT/Firewall.

# SIP Devices Support and Workarounds

Many currently available SIP devices and applications incorrectly implement various aspects of the SIP protocol.
The CommuniGate Pro SIP Module tries to compensate for certain client problems and bugs, based on the type of SIP devices connected to it.

Use the WebAdmin Interface to configure the SIP workarounds. Open the Real-Time pages in the Settings realm, then open the SIP pages. Click the Workarounds link. The Workarounds table appears:

| Problems/Bugs | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Agent Name | Microsoft | SubPresence | FixContacts | NoTCP | NoMaddr | NoPath | BadByeAuth | NeedsEpid | NoSubMWI | ActiveHold | TCPPing |
| | | | | | | | | | | | |

To specify workarounds for a certain product, put the product name into the last (empty) table element, select the required workarounds and click the Update button.

To remove a certain product, remove its name from the table, and click the Update button.

A similar table exists for remote sites:

| Problems/Bugs | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Agent Name | Microsoft | SubPresence | FixContacts | NoTCP | NoMaddr | NoPath | BadByeAuth | NeedsEpid | NoSubMWI | ActiveHold | TCPPing |
| | | | | | | | | | | | |

When the SIP module is about to relay a Signal request to a remote destination, it applied the workaround methods specified for the request URI domain as well as the methods specified for the target URI domain.

The currently implemented workaround methods are:

```
Microsoft
```
The entity is a Microsoft client. Protocol messages are signed, and other SIP protocol derivations are processed.

```
SubPresence
```
The entity supports Presence, but does not implement a push-type Presence Agent (`Publish`). The Server will send `SUBSCRIBE` requests to monitor the entity Presence status.

```
noTCP,noMaddr
```
The entity does not support `transport` and/or `maddr` Contact parameters. The Server will modify the Contact data sent to this entity.

```
noPath
```
The entity does not support the RFC3327 (`Path` fields).
The Server will modify the Contact data sent to this entity.

```
badByeAuth
```

The entity incorrectly calculates Authentication digests for non-INVITE (`BYE`, `NOTIFY`, `REFER`) requests.

`needsEpid`

The entity uses a non-standard `epid=` parameter in its From/To URIs and fails to work if the peer does not preserve this non-standard parameter.

`NoSubMWI`

The entity supports the Message-summary Event package (to implement MWI - Message Waiting Indicator), but it fails to send SUBSCRIBE requests to activate this service.
The Server will subscribe the client on registration.

`TCPPing`

When this entity sends a REGISTER request over a TCP connection from a NAT'ed network, do not start to send the PING packets to that entity. Instead, enable the "Keep Alive" option for the request TCP connection.
**Note:** most OS have very long time-outs for the "Keep Alive" option. If you plan to use this workaround, it is recommended to decrease these timeouts in the Server OS to 1-3 minutes.

`badUpdate`

The entity advertises support for the SIP UPDATE method, but it incorrectly processes SIP UPDATE requests.

The following Web Site contains a periodically updated document listing the tested SIP Clients, the problems discovered and the known workarounds.

# Routing

The SIP module immediately (on the first Router call) accepts all signal addresses with IP-address domains, i.e. with domain names like `[xx.yy.zz.tt]`. Please note that the Router adds brackets to the IP-address domain names that do not have them, and the Router changes the IP addresses of local domains to those domain names. The Router performs these operations before calling the modules.

On the final call, the SIP module accepts signal to any domain if that domain name contains at least one dot (`.`) symbol. If the Relay via option is selected, all these addresses are rerouted to the specified Relay via domain.

Before accepting an address, the SIP module checks if the address does not contain any `@` symbol, but contains one or several `%` symbols. In this case, the rightmost `%` symbol is changed to the `@` symbol.

If the target domain name contains a `._udp`, `._tcp`, or `._tls` suffix, the corresponding transport protocol is used, and the suffix is removed from the target domain name.

# Monitoring SIP Activity

The Monitors realm of the WebAdmin Interface allows Server Administrators to monitor the SIP module activity. The SIP Monitor page contains two frames - the receiving (Server) frame, and the sending (Client) frame.

The SIPS frame displays the active SIP Server transactions:

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | 2 of 2 selected |
| **ID** | **Status** | **Msg** | **Phase** | **Source** | **Target** | |
| 38984 | waiting (15s) | 0 | completed | udp[10.10.0.226:59645] | SIGNAL-40726 | OPTIONS sip:ns.communigate.com |
| 38994 | waiting (7s) | 0 | completed | tcp[10.0.14.193:24777] | SIGNAL-40734 | MESSAGE sips:user@communigate.com |

The SIPC frame displays the active SIP Client transactions:

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | 2 of 2 selected |
| **ID** | **Status** | **Msg** | **Phase** | **Source** | **Target** | |
| 35184 | waiting (2m) | 0 | provisioned | SIGNAL-40726 | udp[10.0.1.34:5060] | INVITE sip:user@10.0.1.34:5060 |
| 35188 | waiting (7s) | 0 | request sent | SIGNAL-40732 | udp[10.0.1.34:5060] | MESSAGE sips:user@communigate.com |

Introduction

Installation

SysAdmin

Network

Objects

Storage

E-mail

Real-Time

Access

Services

Directory

Clusters

Applications

Miscellaneous

Licensing

WebMail

Pronto!

PBX

IM &
Presence

Overview Rules SIP XMPP SMPP PSTN NAT Media Parlay

# XMPP Module

- **Extensible Messaging and Presence Protocol (XMPP)**
- **Server Settings**
- **Client Settings**
    - **Secure (encrypted) Relaying**
- **Monitoring XMPP Activity**
- **Registration**
- **Chat Rooms**
- **Components**
- **Supplementary Discovery Items**

The CommuniGate Pro XMPP Module implements the XMPP protocol via TCP/IP networks.

The XMPP module implements the client-server XMPP protocol. The module allows end-user applications (XMPP clients) to log into the CommuniGate Pro Server and to execute Real-Time Signal operations.

The XMPP module implements the server-server XMPP protocol. The module allows the remote XMPP systems (remote servers) to connect to the CommuniGate Pro Server, and it allows the CommuniGate Pro Server to connect to the remote XMPP systems, so they can exchange Real-Time Signal requests and responses.

The module implements the basic XMPP protocol and its extensions (XEPs).

# Extensible Messaging and Presence Protocol (XMPP)

The CommuniGate Pro XMPP Module implements the XMPP protocol functionality. It uses one TCP Listener for both client-server and server-server connections, distinguishing them not by the port number the remote side connects to, but by the XML data transferred.
By default, the XMPP Module TCP Listener uses the plain-text ports 5222 and 5269 and the secure (TLS) port 5223.

When a client-server connection is established, the XMPP module authenticates a user, and creates a session for the user Account. It then sends the Real-Time Signals on that Account behalf.

When a server-server connection is established, the XMPP module receives XMPP requests, converts them into internal Real-Time Signals, and submits them to the Real-Time component for processing and further delivery. It also receives the Signals directed to it by the Real-Time component and delivers them to remote XMPP systems.

The XMPP Module maintains a separate queue for each target and source domain, i.e. there are different queues for requests to the target.dom Domain coming from addresses in the source1.dom and source2.dom domains.
The XMPP module tries to open a new TCP/IP connection for each queue. If a connection is established, it sends all enqueued requests (as XMPP XML stanzas). When all enqueued requests are sent, the module keeps the connection open waiting for new requests to be sent to this queue. If the connection stays idle for the specified period of time, the

XMPP module closes it.

The XMPP `message` requests are sent as `MESSAGE` Real-Time requests, the XMPP `presence` requests are sent as `SUBSCRIBE` and `NOTIFY` Real-Time requests (using the `presence` Event package), the XMPP `iq` requests are sent as `INFO` Real-Time requests.
The XMPP request data beyond the basic information is sent as the `P-XMPP-Data` field data.

# XMPP Server Settings

Use the WebAdmin Interface to configure the XMPP module. Open the Real-Time pages page in the Settings realm, then open the XMPP pages.
Click the Receiving link to open the XMPP Server Settings.

| Processing | | | |
|---|---|---|---|
| Log Level:  Major & Failures | **Listener** | Channels:  100 | |

Use the Log Level setting to specify what kind of information the XMPP module should put in the Server Log. Usually you should use the `Major` (message transfer reports) or `Problems` (message transfer and non-fatal errors) levels. But when you experience problems with the XMPP module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.

The XMPP server records in the System Log are marked with the `XMPPI` tag.
If the remote party specifies that the connection is used for a client session (as opposed to server-to-server transfer), the System Log tag is changed to `XMPPS`.

System Log records for outgoing server-to-server XMPP connections are marked with the `XMPPO` tag.

When you specify a non-zero value for the `Channels` setting, the XMPP module creates a TCP listener, and the module starts to accept XMPP connections from client applications and remote servers.
The setting is used to limit the number of simultaneous connections the XMPP module can accept. If there are too many incoming connections open, the module will reject new connections, and client applications and remote servers should retry later.

By default, the XMPP module Listener accepts clear text connections on the TCP port 5222 ("client port") and 5269 ("server port") and secure TLS connections on the TCP port 5223 ("secure client port").
Follow the listener link to tune the XMPP Listener.

The XMPP module supports the `starttls` command; it allows client applications and remote servers to establish a connection in the clear text mode and then turn it into a secure connection.

The XMPP module supports all available SASL authentication methods. Additionally, the module supports the legacy Jabber authentication - plain text "password" and "digest".
To disable the "digest" Jabber authentication method, you need to disable the CRAM-MD5 method for the target Domain.

# XMPP Client Settings

Use the WebAdmin Interface to configure the XMPP module. Open the Real-Time pages page in the Settings realm, then open the XMPP pages.

Click the Sending link to open the XMPP Client Settings.

| Processing | |
|---|---|
| Log Level:    Major & Failures | Idle Timeout:    2 min |

Log Level
> This is the same setting as seen on the XMPP Server Settings page.

Source IP Address
> This option selects the default *source network address* for outgoing XMPP connections. You can allow the server OS to select the proper address or your can explicitly select one of the server IP addresses as the default source network address.

Use Domain IP Addresses
> This option selects *source network addresses* for outgoing XMPP connections. If this option is selected, the XMPP Module will use the first Assigned IP Address of the sender's Domain, if the Domain Assigned IP Addresses can be used for outgoing connections.
> If this option is not selected, or if the sender's Domain does not have any Assigned IP Address, the XMPP Module uses the default *source network address*.

Idle Timeout
> Use this setting to specify how long the XMPP module should keep an outgoing connection open if it has no data to be sent to a remote server.

# Secure (encrypted) Relaying

You can configure your CommuniGate Pro Server XMPP module to use secure (encrypted) connections when relaying IM and presence data to certain remote sites. This feature is especially useful if your company has several offices and the traffic between the offices is sent via the public Internet.

You should simply list the domain names that should receive IM/Presence information from your server via secure connections:

| Send Encrypted (SSL/TLS) |
|---|
| to Domains: (high security) |
| wherever possible (low security) |

The specified names can contain a wildcard - the asterisk (*) symbol.

When the CommuniGate Pro XMPP module connects to a relay of one of the listed domains, it checks if that relay supports the starttls protocol command. Then the XMPP module uses this command to initiate a secure connection with that relay.

The CommuniGate Pro XMPP module checks the validity of the remote relay Certificate using the specified set of the Trusted Certificates.
The remote relay Certificate *subject* must contain the cn (*Common Name*) field that matches either the domain name of the remote site, or the name of this relay. This can often cause a problem, since the domain company.dom may have the SRV record xmpp.company.dom, but the computer with the xmpp.company.dom address has the "main" DNS name server.company.dom and its Certificate is issued to that name (its Certificate *subject* contains server.company.dom in the cn field).

To solve this problem, you should explicitly route all traffic to the company.dom domain via the server.company.dom

relay, using the following [Router](#) record:

```
NoRelay:company.dom = company.dom@server.company.dom._via
```
**Note:** this feature ensures that information sent from your server to the remote relay is transferred securely. To provide complete end-to-end security, you should verify that:

* users submit data to servers either using a private network, or using TLS/SSL connections over the public Internet;
* all XMPP/SIP servers and relays exchange messages either using a private network, or using TLS/SSL connections over the public Internet;
* users receive data either using a private network, or using TLS/SSL connections over the public Internet.

If the domain is listed in the Send Secure To Domains list, and the receiving server does not support the `starttls` command, or the remote server certificate cannot be validated, or the remote server certificate Subject does not match the domain or domain relay name, all Signals sent to that domain are **rejected**, ensuring that no data is sent via a potentially insecure link.

`wherever possible`
> Select this option if you want the XMPP module to try to use SSL/TLS connections with all remote XMPP servers that support this feature. If the remote domain is **not** listed in the Send Secure To Domains list, but the remote server supports the `starttls` command, the XMPP module tries to establish a secure (SSL/TLS) connection with that server.
> The module does not check the remote server Certificate validity or the Certificate Subject in this case. If the `starttls` command or secure connection negotiations fail, the server defaults back to plain-text communication and sends data via an unencrypted channel.
> Some servers advertise `starttls` support, but fail to accept SSL/TLS connections. When this option is selected, it is impossible to send Signals to those servers. To solve this problem, inform the broken server administrator, and enter the server domain into the Send Secure To Domains list, prefixed with the exclamation point (`!`) symbol. The XMPP module will not try to use SSL/TLS connections with that server/domain.

---

# Monitoring XMPP Activity

The Monitors realm of the [WebAdmin Interface](#) allows Server Administrators to monitor the XMPP module activity.

Click the Real-time link in the Monitors realm to open the XMPP Monitoring page. This page has the same format as the [IMAP Monitoring](#) WebAdmin page.

---

# Registration

The XMPP module implements the XEP-0077 extension (registration and password changes).

The [Auto Signup](#) option allows users to 'register' (to create Accounts for themselves).

If the [Allow to Modify](#) Password option is enabled, the users can modify their passwords themselves.

---

# Chat Rooms

The XEP-0045 extension (Multi-User Chat) is implemented outside the XMPP module itself, using the [CG/PL](#) `chatroom` [Named Task](#).

The Named Task is started automatically when the first user tries to access it. The Named Task receives all Instant Message and Presence requests sent to it from various clients (XMPP, XIMSS, other Tasks, etc.), and distributes these Signal requests to the room participants.

# Components

The XMPP module implements the XEP-0114 extension (Jabber Component Protocol). It allows "trusted components" to connect to the CommuniGate Pro XMPP module. Usually these "trusted components" are external servers acting as gateways to some other IM/Presence networks.

Use the WebAdmin Interface to open the Receiving page in of the XMPP Module Settings.

| Protocol |
|---|
| Component Password: |

`Component Password`
> Use this field to set a password ("shared secret") string. When "trusted components" connect to the XMPP module, they must present this password in order to establish a connection.

The "trusted components" specify the domain name they are serving. The XMPP Module accepts all Signal addresses routed to any domain specified by any "trusted component" and it relays these Signals to the "trusted component" via the extablished XMPP connection.
If, for example, a "trsuted component" has connected to the XMPP Module to serve the `icq.company.dom` domain, then all Signal requests directed to any address in the icq.company.dom domain is accepted with the XMPP module, and it delivers the Signal request to this "trusted component" (which is likely to be a gateway to the ICQ network).

# Supplementary Discovery Items

The XMPP module allows you to specify additional XMPP entities (domains, JIDs, resources) that are returned when the 'item discovery' request is sent to the Server or to any of the Server Domains. These items are returned together with the list of all chatrooms and registered Components.

You may want to use this feature to add the commonly-used external Gateways and Chatrooms to the 'item discovery' lists retrieved with the XMPP and XIMSS clients.

Use the WebAdmin Interface to open the Receiving page in of the XMPP Module Settings.

| Supplementary 'Discovery' Items |
|---|
|  |

To add a new item, enter its name into the last, empty field and click the Update button.
To remove an item, delete its name, and click the Update button.

CommuniGate **Pro**

**Introduction**
**Installation**
**SysAdmin**
**Network**
**Objects**
**Storage**
**E-mail**
**Real-Time**
**Access**
**Services**
**Directory**
**Clusters**
**Applications**
**Miscellaneous**
**Licensing**

**WebMail**
**Pronto!**
**PBX**

# SMPP Module

- **Short Message Peer to Peer Protocol (SMPP)**
- **Configuring the SMPP Module**
- **Configuring SMPP Server Records**
- **Routing Signal Requests to the SMPP Module**
- **SMS Dialogs**

The CommuniGate Pro SMPP Module implements the SMPP protocol via TCP/IP networks.

The SMPP module implements the client (ESME) part of the SMPP protocol. The module can connect to the specified SMPP servers (SMSC, Short Message Service Centre) and relay IM (Instant Message) Real-Time Signal requests as SMS messages to recipient mobile phones.

The SMPP module can also receive SMS messages from SMPP servers and send them as IM Signal requests to the specified recipients.

# Short Message Peer to Peer Protocol (SMPP)

The CommuniGate Pro SMPP Module implements the SMPP protocol functionality. When the Server Administrator specifies one or several SMPP servers (SMSC systems), the SMPP Module opens a TCP connection to the specified server address and "binds" using the specified credentials.
When an IM Signal is routed to the SMPP Module, the Module finds the SMPP server it is routed to, and sends it to that server via the SMPP connection.

To maintain a connection when it is idle (no IM Requests to relay), the module periodically sends the `enquire_link` request to the SMPP server.

If a connection to the server cannot be established, or it has been broken and it is being re-established, the IM Signals are enqueued. They are relayed to the server when a connection is established.

The SMPP module processes only the `MESSAGE` Real-Time requests, all other requests are rejected.

# Configuring the SMPP Module

Use the WebAdmin Interface to configure the SMPP module. Open the Real-Time pages page in the Settings realm, then open the SMPP page.

| Processing | | | |
|---|---|---|---|
| Log Level:   Major & Failures | Reject if not Relayed in:   30 sec | Dialog Timeout:   20 min | Source IP Address:   OS default |

`Log`

Use the Log Level setting to specify what kind of information the SMPP module should put in the Server Log. Usually you should use the `Major` (Signal relay reports) or `Problems` (relay reports and non-fatal errors) levels. But when you experience problems with the SMPP module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.

The SMPP records in the System Log are marked with the `SMPP` tag.

**Reject if not Relayed in**
> If a Signal request to be relayed to a SMPP server is enqueued longer than this period of time, the Signal request is rejected.

**Dialog Timeout**
> This option specifies the [SMS Dialog](#) binding time-out.

**Source IP Address**
> This option selects the *source network address* for outgoing SMPP connections. You can allow the CommuniGate Pro server OS to select the proper address or your can explicitly select one of the server IP addresses.

---

# Configuring SMPP Server Records

You can specify one or several SMPP servers the SMPP module should connect to.

| Name | Mode | SMSC Address | System ID | Password | Sender | Recipient | Dialogs | Encoding | Ping Period |
|------|------|--------------|-----------|----------|--------|-----------|---------|----------|-------------|
| | Send | | | | | | | gsm-03.38 | 5 min |
| | Send & Receive | | | | | | | Western European (ISO) | 30 sec |
| | Off | | | | | | | gsm-03.38 | Never |

**Name**
> This is an internal name of this SMPP server configuration. Any name can be used, but there should be no two SMPP server configurations with the same name. An empty name is allowed.

**Mode**
> The operation mode for this SMPP server configuration: send-only, receive-only, send-and-receive. The SMPP module logs into the SMPP server as "Transmitter", "Receiver", or "Transceiver".
> To disable the SMPP server configuration without removing it, select the Off mode.

**SMSC Address**
> The SMPP server (SMSC) domain name or Network IP Address, with the port number to connect to, separated using the colon (`:`) symbol. If the port number is not specified, the port 2775 is used.
> If the connection to the SMPP server should be established securely (using the TLS protocol), add the `tls:` prefix before the server domain name.

**System ID, Password**
> These fields contain the credentials to be used to "bind" (log into) the SMPP server.
> By default the SMPP module uses the `CommuniGate` string as SMPP "System Type". Some SMPP servers require some other string to be used as "System Type". You can specify that string in the System ID field, after the actual System ID, separating it with the `@` symbol. For example, if the System ID is user1234, and the requires System Type is SMPP, then specify the `user1234@SMPP` value in the System ID field.

**Sender**
> This field is used to compose the "sender" for the SMS messages sent. Usually it is a phone number assigned to the SMPP account you bind to. Use the `+countryCodephoneNumber` format to specify the "international" sender format.
> If this field contains the `*` symbol, the actual sender E-mail address is substituted. If this address is `+number@telnum`, only the *number* part is used. If this address is `userName@null`, only the *userName* part is used.
> If this field is empty, it is processed as if it has the `*` value.
> If the resulting address starts with the `+` symbol, it is removed and the address is sent to the SMPP server as an "international" one.
> Otherwise it is sent either as a "local" one (if it starts with a digit) or as an "alphanumeric" one.

**Dialogs**
> When this option is selected, messages sent and received via this SMPP configuration will be processed to support [SMS Dialogs](#).

**Recipient**
> An empty string or an E-mail address of the default recipient. If specified, incoming messages are sent to the specified address if:

> - the Dialogs option is disabled
> - the Dialogs option is enabled, but no address for the incoming message sender has been remembered (no "binding" found)
> - the Dialogs option is enabled, and a "binding" address is found, but the incoming message has the `@@` prefix (two "at" symbols and a space)

> If this setting is not specified (left blank):

- if the Dialogs option is enabled, but no address for an incoming message sender has been remembered (no "binding" found), that incoming message is rejected.
- if the Dialogs option is disabled, a message is routed to the destination address specified with the SMPP protocol: if a message is directed to the phone number `+14155551212`, then it is sent to the +14155551212@telnum address, which can be routed to some local Account (via the Telnum or other mechanism).

`Encoding`
The preferred character set (encoding) to use for message sending. If a message cannot be encoded using this character set, it is sent using the Unicode encoding.

`Keep Alive`
This option specifies the maximum time period between commands sent to the SMPP server.
If there is no message to send to the SMPP server within this time period, the `enquire_link` command is sent to prevent the server from closing the connection.

To create a new SMPP server configuration, enter the server name and other settings into the last (empty) element, and click the Update button.

To remove an SMPP server configuration, enter an empty string into its Server (not the Name!) field, and click the Update button.

---

# Routing Signal Requests to the SMPP Module

To route a Signal to the SMPP module:

- route it to the `phonenumber@smpp` address, where *phonenumber* is the target telephone number in the E.164 format (with the leading + sign). The Signal request will be sent to the first available (enabled) SMPP server configured. If there is an enabled SMPP server with an empty name, the Signal will be sent to that server.
- route it to the `phonenumber@name._smpp` address, where *name* is the internal name of a configured and enabled SMPP server.

Example:
with the SMPP server configuration displayed above, Signals directed to `+14155551212@smpp` will be sent to provider1 (the `64.173.55.161:901` SMPP server), while Signals directed to `+14155551212@masslist._smpp` will be sent to masslist (the `smpp.provider.dom:950` SMPP server).

Usually, SMS Messages are directed to a regular phone number, i.e. to some `+countryCode phoneNumber@telnum` address. By default, this address is routed to the `gatewaycaller` application that relays incoming Instant Messages to one of the SMPP servers. It can take take the SMPP server name from the PSTNSMSGateway setting of the sender, or, if this setting is empty, from the application parameter (used as a default gateway).
To use the provider1 SMPP server as the default one, add the following record to the Router Table:

```
IM:<*@telnum> = gatewaycaller{*,provider1}#pbx
```

---

# SMS Dialogs

This SMPP Module implements "SMS Dialogs": when a CommuniGate Pro user sends an Instant Message to some SMPP endpoint (such as a mobile phone), the endpoint user can reply and the reply will be sent to the CommuniGate Pro user as an SMS.

When an instant message sent from some *user1@domain1* address is relayed as an SMS message to the phone number *XXXXXXX*, the SMPP module adds a prefix to the message text:
`@a=user1@domain1 message text`

Subsequent instant messages from the same *user1@domain1* address are relayed to the *XXXXXXX* phone number with the "short" prefix:
`@a message text`

When an instant message sent from some other *user2@domain2* address is relayed as an SMS message to the same phone number *XXXXXXX*, the SMPP module adds a different prefix to the message text:
`@b=user2@domain2 message text`

Subsequent instant messages from the same *user2@domain2* address are relayed to the *XXXXXXX* phone number with the "short" prefix:
`@b message text`

The SMPP module remembers these "bindings", consisting of the sender address, the destination phone number, and the prefix (a low-case letter) assigned.

When a user of the *XXXXXXX* phone replies to a message, the SMPP module checks all bindings for the *XXXXXXX* phone number. If there is exactly one binding, the module relays the received SMS message as an instant message to the address from that binding.
If no binding is found, or if there are two or more binding for this phone number, the SMS message is not accepted.

The user can reply with an SMS message starting with the `@x:` or `@x_` prefix, where "_" is the space symbol.
In this case the SMPP module tries to find a binding for the *XXXXXXX* phone number and the *x* prefix. If it is found, the SMS message is relayed to the address in the found binding, otherwise the SMS message is not accepted.

Example: single dialog as seen on the mobile phone screen:

```
@a=smith@company.dom Hi! Got
good news about that contract.
                 Great. Signed?
@a They've signed it, we'll get
it tomorrow.
                  Congrats, CU!
```

Example: two dialogs as seen on the mobile phone screen:

```
@a=susan@company.dom Will you be
in town this weekend?
                             Yes
@a Please come visit us on
saturday, we'll be at home all
day long.
@b=smith@company.dom Hey, just
received their signed contract!
    @b good! I'll be there in 30m
        @a Thnx a lot, I will!
```

The SMPP Module retires each "binding" if the there it was not used in either direction for the specified Binding Timeout time period.
If the Binding Timeout value is too low, then a cell phone user may not be able to answer to a recenlty received SMS message, because by the time the user discovers the SMS, the binding could have expired. If the Binding Timeout value is too high, then cell phone users may have several active bindings too often, forcing them to use `@x` prefixes too often.

Introduction

Installation

SysAdmin

Network

Objects

Storage

E-mail

Real-Time

Access

Services

Directory

Clusters

Applications

Miscellaneous

Licensing

WebMail

Pronto!

PBX

| IM & Presence | | | | |
|---|---|---|---|---|
| Overview Rules | | SIP XMPP SMPP **PSTN** NAT Media Parlay | | |

# PSTN

- **Incoming Calls**
- **Remote SIP (RSIP) Registrations**
- **Outgoing Call Routing**
- **Outgoing Calls via B2BUA**
- **Local Area Calls**
- **Domestic Calls**
- **Emergency Calls**
- **PSTN Account Settings**

The CommuniGate Pro Server can use various *PSTN Gateways* to connect the modern VoIP network to the traditional Public Switched Telephone Network (PSTN). While most Gateways are implemented as standards-based SIP devices, they can experience problems when end-users connect to these Gateways directly. The most common problems include:

- PSTN addressing scheme is different from VoIP: traditional numeric (E.164) addresses are used instead of E-mail type account@domain VoIP addresses.
- PSTN is not a free network, and Gateway may require certain authentication data to establish and maintain a call. This information is used for billing.
- PSTN is not a free network, and its call transfer logic is different because of that. As a result, most Gateways do not support call transfer operations.
- some PSTN gateways are designed to support individual devices (such as SIP phones) and require period REGISTER operations to direct incoming calls to those devices.

To solve these problems, CommuniGate Pro provides a set of "stock" Real-Time Applications.

This section explains how these stock applications work, and how they can be configured. While these applications are flexible enough to serve various configurations, Server and Domain Administrators can upload their own, customized versions of these applications.

# Incoming Calls

PSTN Gateways can direct incoming PSTN calls to a SIP entity, such as a CommuniGate Pro Server. These incoming Gateways usually are unable to transfer calls, to switch media streams, etc.

To overcome PSTN Gateway limitations, incoming calls from those gateways are usually directed to a CommuniGate Pro Real-Time applications acting as a B2BUA.
There is a stock `gatewayincoming` application designed to be used for this purpose.

The stock `gatewayincoming` application expects the destination address to be delivered to it as an application parameter. If the Gateway sends incoming requests to your system using addresses in a special domain `incoming.company.dom`, then the following Router record will direct them to the `gatewayincoming` application:

```
<*@incoming.company.dom> = gatewayincoming{*}#pbx@localhost
```

If you know that the Gateway addresses all your local Accounts as "domestic" numbers (numbers without the country code), you can correct his in the Router record. For example if the Gateway uses 10-digit North American numbers without the `1` country code, you can use the following Router record:

```
<*@incoming.company.dom> = gatewayincoming{+1*}#pbx@localhost
```

Incoming PSTN calls are often directed not to a particular user, but to a Real-Time application acting as a "PBX center". This application answers incoming calls, and allows the callers to select the user to call, call the selected Account, and bridges the call. Since this application also acts as a B2BUA, PSTN Gateway calls can be routed to it directly, without first routing them to the special B2BUA application such as the `gatewayincoming` application:

```
<*@incoming.company.dom> = pbx#pbx@localhost
```

# Remote SIP (RSIP) Registrations

Some PSTN Gateways cannot be configured to direct incoming calls to a specific address (such as a CommuniGate Pro Account). Instead, these Gateways expect SIP REGISTER requests to be sent to them periodically.
These SIP requests specify the Gateway target account (usually - a PSTN phone number), the credentials (such as some username and password), and the SIP address to direct incoming PSTN calls to.

Each CommuniGate Pro Account can contain zero, one, or several RSIP records. Each RSIP record specifies an account information for some remote PSTN Gateway. The CommuniGate Pro server periodically sends the SIP REGISTER requests to that gateway using the specified information, and then the PSTN Gateway receives a phone call, it directs it to this CommuniGate Pro Account.

A Server or a Domain Administrator with the RSIP Registrations Access Right can manage the Account RSIP Registration records.
Open the WebAdmin Account Management pages, then open the RSIP page in the Real-Time section:

| Name | Register Every | Account | at Host | Authentication Name | Password | Target | Last | |
|------|----------------|---------|---------|---------------------|----------|--------|------|---|
| localVoip | 3 min | | | | | | 5:32:00AM | |
| AltVoip | 5 min | | | | | | 5:32:50AM | Illegal password |
| | Never | | | | | | | |

Each RSIP record should have a unique name, used solely for its identification with the Account.

To create a new RSIP record, select some name for it and enter it into the `Name` field in the last table row. Fill other fields and press the Update button.

To modify an RSIP record, modify values in the record fields and press the Update button.

To remove an RSIP record, set its Period value to Never and press the Update button.

`Register Every`
> This setting specifies how often the Server should send the SIP REGISTER requests. These requests use the SIP registration expiration time which is 2 minutes longer than this time period.
> Use the value the PSTN Gateway provider recommends.

Account
> This setting specifies the Account name (the address) for the REGISTER requests. If the specified value does not contain the `@` symbol, the `at Host` value is appended to it then the REGISTER request is sent.
> Use the value assigned to you by the PSTN Gateway provider (usually - the phone number).

at Host
> This setting specifies the domain name or the IP Network Address of the PSTN Gateway. The SIP Register requests are sent to that address.
> Use the value recommended by the PSTN Gateway provider.
> If the REGISTER request should be sent via some SIP proxy, specify that proxy domain name or the IP Network Address after the PSTN Gateway name, separated with the `@` symbol: *gatewayName@proxyName*

Authentication Name, Password
> These settings specify the credentials (the user name and the password) needed to authenticate the SIP Register request sender.
> **Note:** the Authentication name may or may not contain the `@` symbol followed by some domain name.
> If the Authentication Name setting is left empty, the Account setting value is used.
> Use the values assigned to you by the PSTN Gateway provider.

Target
> This setting specifies the Account address the PSTN Gateway should send the incoming calls to. Usually this field is left empty, in this case the calls are directed to this Account.

The last field contains the time when the last SIP REGISTER request completed, and, optionally, an error code received from the PSTN Gateway.

The RSIP Registration functionality is implemented using the [Chronos](#) component. At the scheduled time, the component starts the `rsipRegister` [Real-Time Application](#), which performs the REGISTER transaction.

The SIP REGISTER request `Contact` field is composed so that incoming calls from the PSTN Gateway are sent to the `gatewayincoming` application which acts as a B2BUA and bridges the call to this Account (or to the Account specified with the Target field).
If you want to send the incoming calls directly to this Account, bypassing the B2BUA, add the asterisk (`*`) symbol in front of the At Host setting value.

When the CommuniGate Pro Server is installed, it creates the `pbx` Account in the Main Domain. A Signal Rule is automatically created for this Account to direct all incoming calls to the [PBX Center](#) application.
If you want to specify SIP Registration records for this `pbx` Account, so it will receive incoming calls from the `sip.provider.dom` PSTN Gateway, specify the At Host setting as `*sip.provider.dom`.

Account users can review their RSIP Registration records and (if the RSIP Registration setting is enabled) modify them using the WebUser Interface.

---

# Outgoing Call Routing

The CommuniGate Pro [Router](#) processes every Signal address in the system.

- The Router uses Default or Custom records to detect telephony (PSTN) type addresses and to convert them to the standard E.164 form (+*country_code area_code local_number*) in the fictitious `telnum` domain name. For example, when a SIP phone user dials `011 44 3335555`, the Server receives that address as `sip:011443335555@client1.dom` URI, and the Router converts this address into +44333555@telnum address. Usually all numeric addresses of the certain length are processed as PSTN numbers.
  See the [Router](#) section to learn about the Router records used to detect telephony type addresses.

- The Router maps PSTN numbers to local or remote VoIP addresses, if possible.
  The Router checks if an address (number) in the fictitious `telnum` domain is assigned to some local Account. If an Account is found, the Router routes the address to that Account.
  For addresses not assigned to any local Account, the Router can use global and local ENUM services, as well as CommuniGate Pro Forwarders.
- The Router directs unmapped PSTN numbers either directly to a PSTN Gateway, or to a CommuniGate Pro Real-Time application designed to support external PSTN Gateways.

# Outgoing Calls via B2BUA

The [Router](#) can route calls to the PSTN network via a [Real-Time Application](#).
The CommuniGate Pro Server comes with a "stock" `gatewaycaller` application that can be used for relaying outgoing PSTN calls to one or several PSTN gateways. This section describes this application functionality.

When the `gatewayCaller` application receives an incoming call, it requires authentication. As a result, only the users with the Accounts on your CommuniGate Pro Server or Cluster can place calls via this application.

When an authenticated call request is received, the application retrieves and uses the following Account [PSTN Settings](#):

`PSTNGatewayName` (Gateway Name)
> The Gateway domain name. It is used as an address to send the call requests to.
> If this setting value is empty, the Account is not allowed to make PSTN calls and its calls are rejected.

`PSTNGatewayVia` (Gateway Address)
> This setting is a non-empty string if the requests to the Gateway should not be sent directly (to the DNS-resolved addresses), but they should be sent via a different proxy. This setting should contain the domain name or the IP address of that proxy.

`PSTNFromName` (Caller ID)
> This setting specifies the address (usually, a PSTN number) to be used as the Caller ID (the `From:` address) in the call request sent to the Gateway. If this setting value is empty, the From: address of the original request is used.

`PSTNGatewayAuthName` (Name for Gateway)
`PSTNGatewayPassword` (Password for Gateway)
> If the Gateway requires authentication, these settings specify the authentication data to use with the requests sent to the Gateway.

`PSTNBillingPlan` (Billing Plan)
> This setting contains the name of the billing plan to apply. The stock `gatewaycaller` application tries to read the specification for the billing plan `standard` from the PBX environment file with the name `billingplan-standard.objdata`. The file should contain a dictionary where each key is a phone number prefix and the value is either a number with the call cost per a minute, or a reason string to use to block calls to numbers with this prefix:

```
{
  1990 = blocked;  // calls to +1-990-XXX-XXXX are blocked
  1 = #0;          // calls to North American numbers are free
  "" = #10;        // everything else costs 10 units per minute
}
```

A CommuniGate Pro Account can use a default value for each of its setting (taken either from the Domain-wide or

Server-wide/Cluster-wide Default Settings) or a setting value can set explicitly for that Account. As a result, all Accounts in all Domains can use the same Gateway and same Gateway credentials, but different Caller ID settings, or each Domain and/or each Account can use its own Gateway, with Domain or Account-specific credentials.

The application needs to retrieve Settings from other Accounts, so it should be started on behalf of an Account with the Domain Administrator access rights.
The automatically created `postmaster` or `pbx` Accounts can be used.

The application expects to get the destination number as its parameter.
The following Router record routes all addresses in the fictitious `pstn` domain to the `gatewaycaller` application started on behalf of the `pbx` Account in the Main Domain, and it passes the "user part" of the `pstn` domain address as the application parameter:

```
<*@pstn> = gatewaycaller{*}#pbx
```

After processing all Account Settings, the application starts a new Task and instructs that Task to send a call request to the selected Gateway using the retrieved Account settings. If the call succeeds, the original and new Tasks "bridge" the media streams.
The original Task processes Signal requests from the caller, the second Task processes the requests from the Gateway. Some of these requests are processed with the Tasks themselves, some are relayed to the peering Task for relaying to the caller or to the gateway.

The technology used to implement this type of call setup (when a call is established using two formally independent Signaling sessions) is called a Back-to-Back User Agent (B2BUA).

When a caller wants to Transfer a call, the Transfer request is sent to the original Task. This Task implements the requested call transfer itself, switching to some other VoIP device/entity. The second Task does not send any Transfer request to the Gateway, but it may send a call update (SIP re-INVITE or UPDATE) request to the gateway in order to redirect its media streams to the new call participant.

Some PSTN Gateways do not support even the simple call update requests. Specify the name of those Gateways with a leading hash (#) symbol. The application will remove that symbol from the name, and it will not bridge media streams. The application will use a CommuniGate Pro Media Server channel to relay media between the caller and gateway. If the caller switches its media address by sending a call Update request, or if the caller sends a call Transfer request, the Media Server channel updates its internal information, and no request is sent to the Gateway, which continues to exchange media data with the Media Server channel.

If you need to use several different gateways per user or per domain (depending on the destination), you can specify the gatewaycaller PSTN*xxxxxx* settings as dictionaries. For example:

```
PSTNGatewayName:{gw1=provider1.com;gw2=provder2.com;}
PSTNFromName:{gw1=fromName1;gw2=fromName2;}
```

You can Route different PSTN numbers to different gateways by passing the second parameter to the gatewaycaller application:

```
<+46(5-12d)@pstn> = gatewaycaller{+46*,gw1}#pbx
<*@pstn>          = gatewaycaller{*,gw2}#pbx
```

All calls to unprocessed numbers in the `pstn` domain will be routed to the gatewaycaller application, and this application will get the second parameter of "gw1" if the callee is in Sweden (a E.164 number starting with +46), and "gw2" in all other cases.
The gatewaycaller application reads all required PSTN*xxxxxx* settings. If a settings value is a dictionary, it takes either the "gw1" or the "gw2" element from the dictionary.

# Local Area Calls

When a user dials a "short" number (a number without the country code and the area code), the call is expected to be delivered to the that number with the user's own area code.

A Router record created during a CommuniGate Pro Server installation routes all calls to 7 digit number address in any local Domain to the `localAreaCall` application.

The stock `localAreaCall` application requires the caller to authenticate, and then it retrieves the following Account [PSTN Settings](#):

`PSTNAreaCode` (Local Area Code)
    This setting specifies the Account country code and the local area code.


The applications needs to retrieve Settings from other Accounts, so it should be started on behalf of an Account with the Domain Administrator access rights.
The automatically created `postmaster` or `pbx` Accounts can be used.

The application expects to get the dialed destination number as its parameter.
The following Router record routes all 7-digit addresses in all local Domains to the `localAreaCall` application started on behalf of the `pbx` Account in the Main Domain, and it passes that address (a 7-digit number) as the application parameter:

```
<(7d)@*> = localAreaCall{*}#pbx@localhost
```

The application retrieves the callers country code and the local area code from the caller's PSTNAreaCode Account Settings, removes all non-digit symbols from the Setting value, catenates the "cleaned" with the dialed number and the `@telnum` domain name, and Redirects the call request to the resulting address.

The [Signal](#) module processes the redirected request using the Router, so it can be directed to a PSTN Gateway or (after successful mapping) directly to some VoIP address.

---

# Domestic Calls

Many countries traditionally use separate calling prefixes for out-of-area domestic and international calls. Many countries use `0area_code local_number` numbers for domestic calls, while `00country_code area_code local_number` numbers are used for international calls.

If all CommuniGate Pro users are located in the same country, domestic calls can be properly routed with a single Router Record. For example, if all CommuniGate Pro users are located in Germany (country code `49`, `0` is the out-of-area prefix), the following Router record can be used (note the order of these records):

```
<0(7-20d)@*> = +49*@telnum
<00(8-20d)@*> = +*@telnum
```

You can also use the `localAreaCall` application to properly route Domestic calls. Route all calls with the domestic prefix to the `localAreaCall` application, but specify the second parameter - a string `c`:

```
<0(8-20d)@*> = localAreaCall{*,c}#pbx@localhost
```

The `localAreaCall` application does the same processing as it does for the [Local Area](#) calls. But instead of using the

entire PSTNAreaCode Setting value, it uses only:

- the country code if the setting value is specified as `country_code-area_code` or as `country_code(area_code)` or
- the first digit if this digit is `1` (North America) or `7` (Russia) or
- the first 2 digits

# Emergency Calls

Call to Emergency Response Centers are done by dialing a well-known number, such as `911` in North America or `112` in Western Europe.

Use the Router to redirect calls to these numbers (in any Domain) to the `emergency` application:

```
<911@*> = emergency#pbx@localhost
```

The stock `emergency` application requires the caller to authenticate, and then it retrieves the following Account [PSTN Settings](#):

`PSTNEmergency` (Emergency Code)
    This setting specifies the way to contact the Emergency Response Center for this Account user.

The applications needs to retrieve Settings from other Accounts, so it should be started on behalf of an Account with the Domain Administrator access rights.
The automatically created `postmaster` or `pbx` Accounts can be used.

The application processes the retrieved PSTNEmergency setting value:

- If the retrieved string has the `call=` prefix, then the remaining part of the string is used as the address to redirect the call to.
- If the retrieved string has the `http=` prefix, then application reads the `emergency.settings` from the Real-Time Environment. This file should contain a [dictionary](#).

  The application makes an HTTP request using the `emergency.settings` dictionary as parameters for the [HTTPCall](#) CG/PL operation. The call body is a dictionary with the following elements:

  `userName`
      the authenticated caller's Account Name (accountName@domainName).
  `fromWhom`
      the caller's name (URI) as specified in the Request From: data.
  `areaCode`
      the PSTNEmergency Setting value with the `http=` prefix removed.

  When the application receives an HTTP response, the HTTP response body should contain a [string](#) or an [array](#) with the destination address(es).

The application redirects the call to the specified address(es).

# PSTN Account Settings

PSTN Settings are implemented as [Custom](#) Account Settings. The main WebAdmin Account Management page does not show the Custom Settings with names starting with the `PSTN` prefix. Instead, a link to a special PSTN Settings page

is available on the Account Settings and Account Default Settings pages.

Click the PSTN link to open the PSTN Settings page:

| | |
|---|---|
| **Local Area Code:** | 1(415) |
| **Emergency Code:** | call=911@our.local.dom |
| **Gateway Domain:** | provider.com |
| **Gateway Address:** | personal.provider.com |
| **Caller ID:** | Boston |
| **Name for Gateway:** | dmak |
| **Password for Gateway:** | ******** |
| **Billing Plan:** | |

A Domain Administrator should have the [PSTN Settings](#) Access Right to modify PSTN Settings.

# NAT Traversal

- **NAT Traversal and Media Stream Proxy**
- **Near-End NAT Traversal**
- **Far-End NAT Traversal**
- **Edge Services**
- **NATed Addresses**
- **NAT Systems**
- **Pinger**
- **Media Proxy**

The "original, "basic" VoIP communication model assumes that endpoints can communicate directly, i.e. that all "entities", both clients (phones, softphones, PBX applications) and servers have "real" Internet IP Addresses. In this situation the entities can exchange media data directly, sending media packets (usually using the RTP or T.120 protocols) directly to each other.

The real-life situation is quite different from this model, and media data cannot be sent directly between endpoints. The CommuniGate Pro Server solves this problem by automatically creating Media Proxies and instructing endpoints to send media data to that Media Proxy for relaying.

A Media Proxy is created when:

- one endpoint is connected to the LAN, while the other one is located somewhere within the WAN.
- one endpoint is located behind a remote NAT, while the other one is not located behind the same NAT (see the option below).
- one endpoint is located within an IPv4 network, while the other one is located in the IPv6 network.
- the Signal component explicitly requested creation of a Media Proxy.

Media Proxies are created with the SIP and XIMSS components when a call is being sent to a remote entity.

# NAT Traversal and Media Stream Proxy

The "basic" communication model assumes that endpoints can communicate directly, i.e. that all "entities", both clients (phones, softphones, PBX applications) and servers have "real" Internet IP Addresses. In this situation the Server is needed only to establish a call. Media data and (in case of SIP) in-call signaling requests are sent directly between the endpoints:
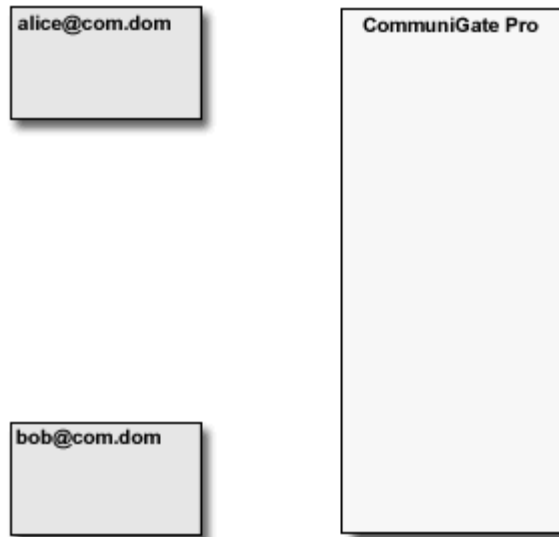
In the real life, many clients are located in remote LANs ("behind a NAT"), or in different LANs so they cannot communicate directly. CommuniGate Pro supports automatic "NAT traversal" for the standard-based Real-Time communications.

---

# Near-End NAT Traversal

The CommuniGate Pro SIP and XIMSS Modules detect the session initiation requests that are sent from one side of NAT to the other side (a request from a LAN client to a party on the Internet/WAN and vice versa). In this case, the Server uses some local server port (or a set of ports depending on the media protocol(s) used) to build a media stream proxy. The Server then modifies the session initiation request to direct the traffic from both sides to that proxy.

The Media Proxy relays media data between the "LAN leg" and the "WAN leg" of the media connection:
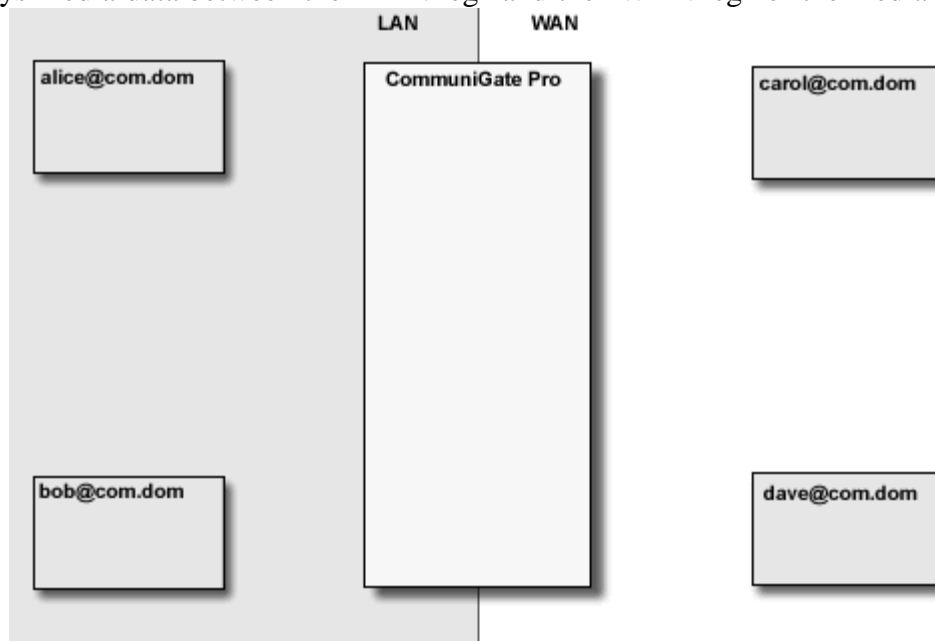


The CommuniGate Pro SIP and XIMSS Modules detect session update (SIP re-INVITE) and session close (SIP BYE) requests and update and remove the Media Proxies accordingly. The time-out mechanism is used to remove "abandoned" Media Proxies.

The CommuniGate Pro provides NAT proxy services for:

* UDP media protocols
* RTP media protocols based on UDP
* TCP media protocols
* T.120 media protocols based on TCP

**Note:** If you need the Media Proxy functionality, make sure that the LAN and NAT data is specified correctly on the LAN IPs and NAT WebAdmin settings pages.

**Note:** The Server automatically builds Media Proxies when it relays requests from IPv4 addresses to IPv6 addresses and vice versa.

---

# Far-End NAT Traversal

The CommuniGate Pro SIP and XIMSS Modules also provide the "far-end" NAT traversal capabilities by detecting requests coming from clients located behind remote Firewall/NATs.
The Modules add appropriate Record-Route and Path headers to these requests and they build Media Proxies to relay traffic to and from those clients.
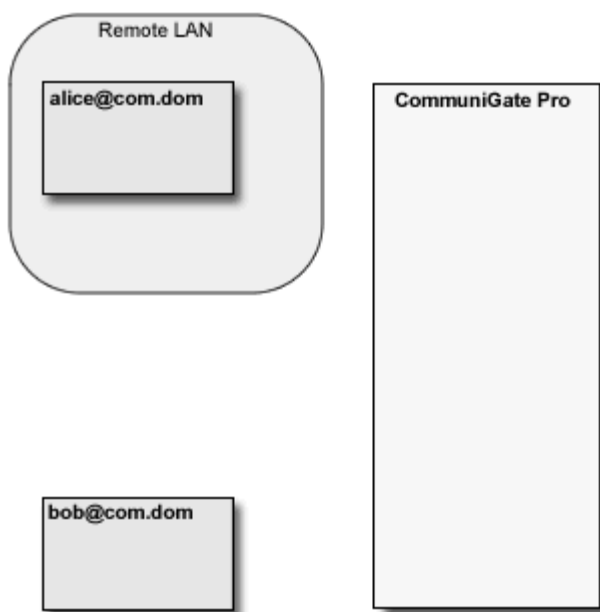


**Note:** modern SIP clients support various NAT traversal methods (STUN, etc.). Many of these implementations are quite buggy, so it is often more reliable to switch the client-side NAT traversal methods off, and rely on the CommuniGate Pro SIP Module far-end NAT traversal capabilities instead.

**Note:** due to the nature of the TCP protocol and the Firewall concept, it is not possible (in general) to open a TCP connection to a client behind a far-end NAT ("near-end" NAT configurations do not have this problem). This means that clients behind a far-end NAT cannot initiate TCP (T.120) sessions.

To solve this problem, you may want to:

* always initiate TCP sessions from a client that is not behind a far-end NAT/Firewall (these clients accept those invitations and start outgoing TCP connections without a problem)
  or
* use an additional copy of the CommuniGate Pro Server on that remote location as a near-end NAT traversal

solution for that network, eliminating a need for far-end NAT traversal on the "main" CommuniGate Pro server
or

- configure the remote Firewall/NAT to relay all incoming TCP request to the T.120 port (usually, the port 1503)
  to a particular workstation behind that Firewall.

# Edge Services

The CommuniGate Pro SIP Module can be used as an "Edge Service" or ALG ("Application Level Gateway"),
providing NAT traversal functionality for users registered on other servers.
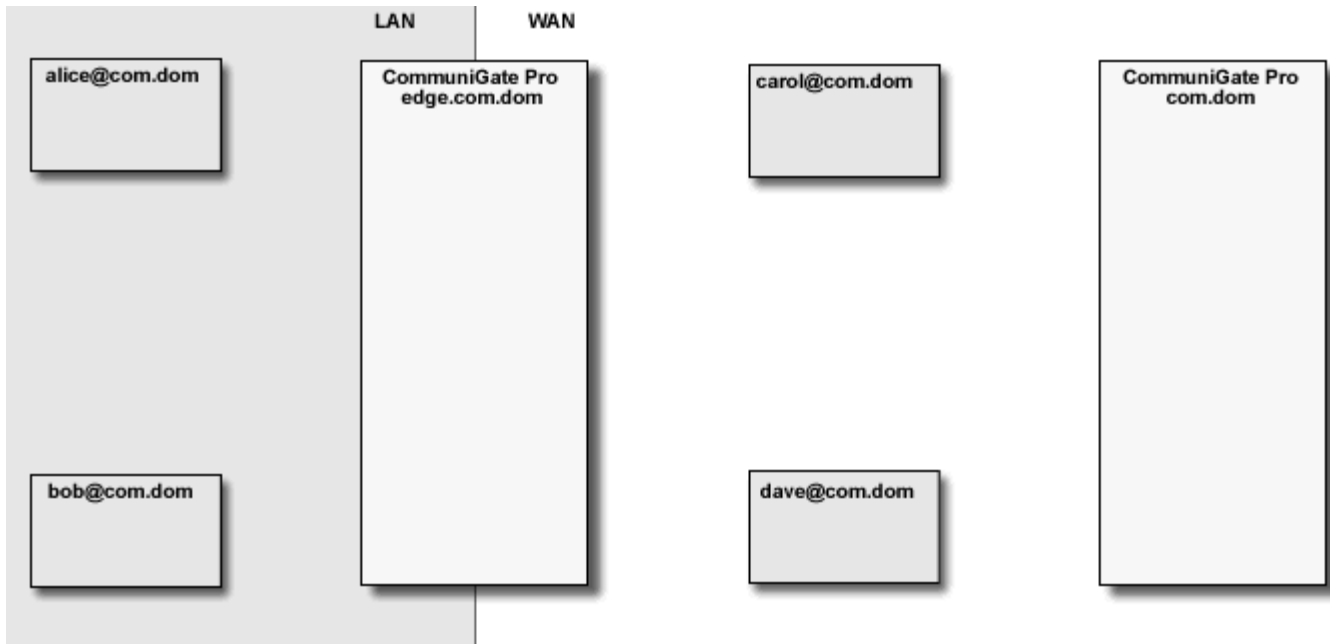
The CommuniGate Pro SIP Module detects "media loops", when a call placed from within LAN is proxied to WAN,
and then proxied back to the same LAN. In this case the Media Proxies are removed, eliminating unnecessary
overhead, and allowing SIP clients to communicate directly within one LAN, while proving registrar services outside
that LAN.
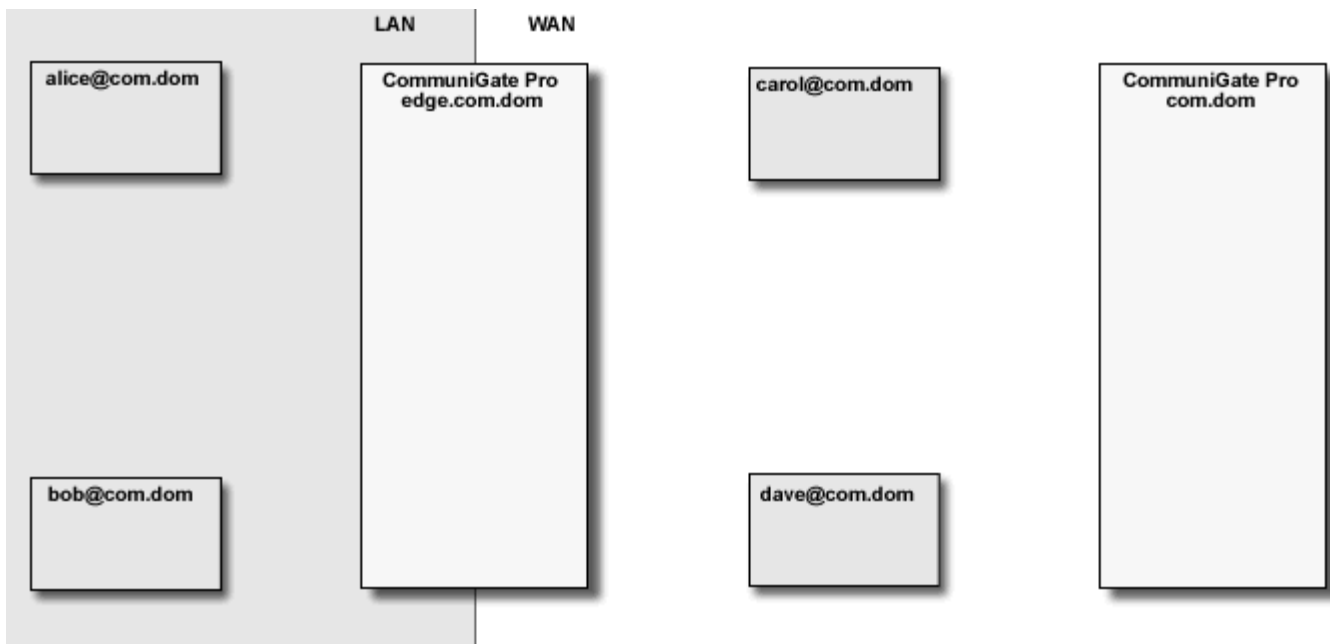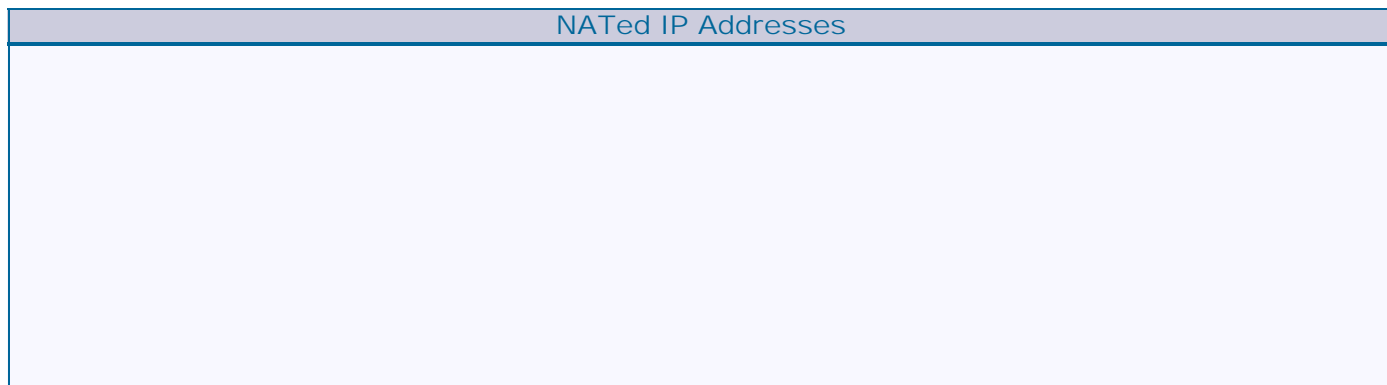
The SIP module can detect much more complex loop cases, either avoiding Media Proxies altogether, or minimizing the number of Media Proxies used.

# NATed Addresses

To detect clients located behind NATs, the Server needs to know which addresses are used on remote networks behind those NATs.
Use the WebAdmin Interface to specify the NATed Addresses. Open the Network pages in the Settings realm, then open the NATed IPs page.
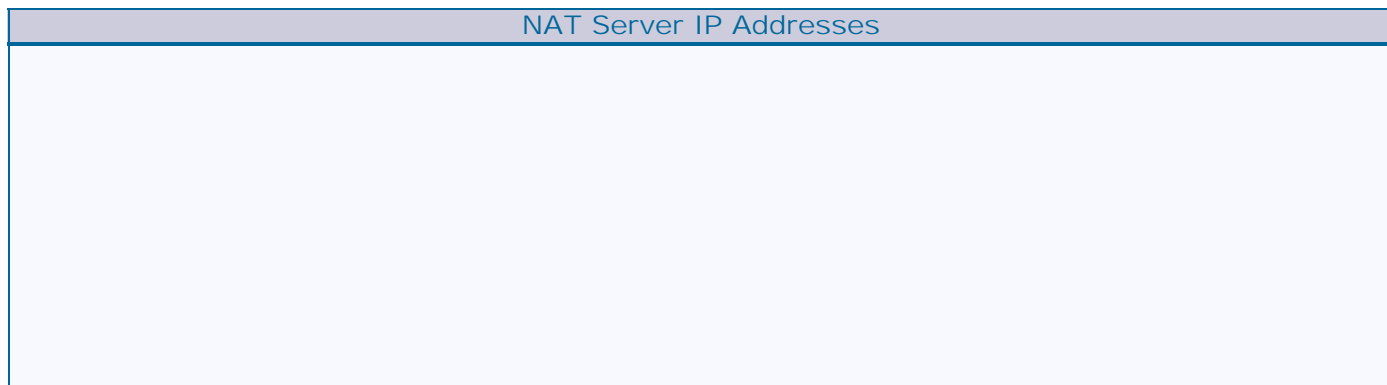
| NATed IP Addresses |
|---|
| |

If a SIP client sends a request to CommuniGate Pro and the client own network address specified in the request headers is included into the NATed IP Addresses list, while the Server has received this request from a different network address, NOT included into the NATed IP Addresses list, the Server decides that this client is behind a NAT.

# NAT Systems

Some NAT servers make attempts to perform "SIP application gateway" functions, changing the IP addresses specified in the relayed SIP requests.
Many of those NAT servers fail to perform these gateway functions correctly, and CommuniGate Pro should treat clients connecting via those NAT servers using its "far-end NAT traversal" functionality, but CommuniGate Pro cannot detect that this functionality is needed, because the client IP addresses specified in their SIP requests are damaged.
You can specify the IP addresses of those broken NAT servers in the NAT Server IP Addresses field: all requests coming from the specified Network Addresses are treated as requests from "NATed" clients:

| NAT Server IP Addresses |
|---|
| |

You may need to relay requests to remote SIP servers (such as PSTN gateways) located behind a far-end NAT. Since these servers do not issue SIP REGISTER requests to your CommuniGate Pro Server, there is no way to automatically

detect these far-end NAT traversal situations.

Include the public Network IP Addresses of these remote SIP servers into the NAT Server IP Addresses field to instruct the CommuniGate Pro SIP Module to use far-end NAT traversal techniques when starting sessions with these SIP servers.

When clients connect to the CommuniGate Pro server from behind multi-homed NAT servers, the client Signal (SIP, XIMSS) connections and media (RTP) streams can come from different IP Addresses.
When a client uses HTTP transactions to connect to WebUser or XIMSS session, the HTTP requests coming from multi-homed NAT servers can come from different IP Addresses.
In thess situations clients can experience lack of media transfer or rejection of session requests caused by a "wrong IP Address" problem.

To avoid this problem, two different IP Addresses are treated as the "same address" if both of them are included into the same IP Address range in the NAT Server IP Addresses list.

# Pinger

To send incoming calls to a SIP client behind a NAT, CommuniGate Pro keeps the "communication hole" between the client and Server open by periodically sending dummy packets to that client.

| NATed Clients Pinger | | | | |
|---|---|---|---|---|
| Log Level: Major & Failures | Clients Limit: 1 | Ping Clients every: | UDP: | minute |
| | | | TCP: | 2 minutes |

`Log Level`
> Use this setting to specify what kind of information the NAT Pinger component should put in the Server Log.
> Usually you should use the `Major` or `Problems` (non-fatal errors) levels.
> The NAT Pinger component records in the System Log are marked with the `NATPING` tag.

`Clients Limit`
> Use this setting to specify how many different NAT clients the Server can ping.

`Ping Clients Every`
> Use this setting to specify how often the Server should send its "pinging" packets to its clients using UDP and TCP protocols.

# Media Proxy

CommuniGate Pro supports various real-time communications. Most of those real-time protocols cannot be used via a NAT/Firewall, so CommuniGate Pro can act as "proxy" for those protocols.

When a client on the LAN tries to communicate with a remote system on the Internet (WAN), CommuniGate Pro creates a Media Proxy - a communication port on its own system.
It forces the client to connect to that Media Proxy instead of the remote system media port.
The CommuniGate Pro Media Proxy communicates with the remote system itself, relaying the data received from the LAN client to the remote system and vice versa.

A Media Proxy is created to serve entries (users) located behind remote NAT devices.

A Media Proxy is created to relay traffic between an IPv4 and IPv6 entries.

Log

Use this setting to specify what kind of information the Media Proxy component should put in the Server Log. Usually you should use the `Major` or `Problems` (non-fatal errors) levels. But when you experience problems with the Proxy component, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well.

The Media Proxy component records in the System Log are marked with the `MEDIAPROXY` tag.

A Media Proxy can create zero, one or several stream proxies for each media stream (for example, one stream proxy for the audio stream, and one - for the video stream). Stream proxy records in the System Log are marked with `UDPPROXY` or the `TCPPROXY` tag.

Source Port Restriction

When this option is selected, the UDP-based media from external non-NATed sources is accepted only when it comes from the correct IP address and port number.

When this option is not selected, only the media source IP address is checked. This helps to serve certain broken devices that send SDP data with incorrectly specifed media port numbers.

UDP TOS Tag

Unless this option is set to `OS default`, the UDP-based media packets get the specified TOS (type of service) tag value. This may help you prioritize the media traffic if your network infrastructure assigned a higher priority to packets with the specified TOS tag.

Relay for clients behind the same NAT

When two endpoints are located behind the same far-end NAT (i.e. when their visible, external Network IP Addresses are the same), this option specifies if a Media Proxy should be built to relay media between these endpoints.

Never

Media Proxy is not built for these endpoints. Use this option when you expect that endpoints behind the same NAT IP can communicate directly.

NAT Sites

Media Proxy is built for these endpoints if their visible Network IP Address is included into the [NAT Server IP Addresses](#) list.

Always

Media Proxy is always built for these endpoints.

**Note**: some remote NAT systems are "multihomed". In this case, a signaling request can come from one external IP Address of that system, while the media stream may come from a different IP Address.

CommuniGate Pro Media Proxies can support these NAT systems if all their external IP Addresses are included into a single IP Address range, and if this IP Address range is included into the [NAT Systems](#) list.

**Overview** **Rules** **IM & Presence** **SIP** **XMPP** **SMPP** **PSTN** **NAT** **Media** **Parlay**

# Media Server

- **Media Server Settings**
- **Codecs**
- **Features**

The CommuniGate Pro Media Server component can *terminate media* for calls - it can play and record audio/video (media) information.

To play media data, the Media Server encodes it using a selected *codec* and sends it to the remote party via the RTP protocol. The Media Server also supports the encrypted SRTP protocol for media exchanges.

To record media data, the Media Server decodes received RTP data, rearranges it to restore the original data frame sequence and converts the resulting data into a standard format.

To participate in a call, the Media Server creates a Media Channel. Each Media Channel uses two UDP ports and sockets for audio data (the AVP/RTP protocol), and, optionally, two UDP ports and sockets for video data.

A Media Channel is created and controlled with PBX Tasks.

A Task can connect its Media Channel to several remote parties, creating a *conference*.

# Media Server Settings

To configure the Media Server component, open the Real-Time pages in the WebAdmin Settings realm, and follow the Media link.

| Processing | | | | | | |
|---|---|---|---|---|---|---|
| Log Level: | Failures | | | UDP TOS Tag: | OS default | |
| Pre-playing: | 60 | msec | | Lingered Playing: | 100 | msec |
| Packet Size: | 20 | msec | | Mixer Delay: | 100 | msec |
| Internal Sampling Rate: | 8000 | Hz | | | | |

Log

Use this setting to specify the type of information the Media Server component should put in the Server Log. Usually you should use the `Failure` (unrecoverable problems only), `Major`, or `Problems` levels. When you experience problems with the Media Server component, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case the media processing internals will be recorded in the System Log. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.

The Media Server component records in the System Log are marked with the `MEDIA` tag.

UDP TOS Tag
>Use this setting to specify the TOS tag for all outgoing UDP packets sent with Media Channels. This tag can be used to set the Media traffic priority on your LAN.

Pre-playing
>When a Media Channel plays a pre-recorded or pre-composed media data, it can initially send some data portions (packets) "in advance". If later network delays cause some packets to arrive to the remote party with a delay, the "pre-played" media data allow that other party to play data smoothly.
>Use this setting to specify the amount of media data to be "pre-played".

Lingered Playing
>PBX Applications often play several media files in a sequence, composing a phrase with several part files. To ensure smooth media parts "merging", a Media Channel reports playing completion when it still has some media data to play. This allows the application to prepare the next media data portion and submit it to the Media Channel for playing.
>The Media Channel starts playing this next portion as soon as it actually finishes playing the previous one.
>Use this setting to specify when Media Channels should report playing completion.

Packet Size
>Use this setting to specify the size (time length) of each RTP packet the Media Server generates.

Mixer Delay
>Use this setting to specify the initial amount of delay the Media Channel uses in the "mixer" mode. Increasing this value increases the mixer quality for lines with large or variable latency, but it also increases the "playback delay".

Internal Sampling Rate
>Specifies the sampling rate for internal audio processing. Increase this value if you use wide-band codecs.

When the remote peer SDP does not announce the special `telephone-event` codec, the Media Server tries to detect the DTMF signals in the received and decoded audio data. The following settings allow you to fine-tune this process.

| Inband DTMF Discovery | | | |
|---|---|---|---|
| Threshold Level: | 60 | Minimal Duration: | 120    msec |
| Signal/Noise Ratio: | 6 | Harmonics Ratio: | 10 |
| Signal Ratio: | 15 | | |

Threshold Level
>Decreasing this setting value allows the Media Channel to detect lesser quality (lower volume) in-band DTMF, but it may also result in "false-positives", when some incoming sounds are incorrectly interpreted as DTMF signals.

Signal/Noise Ratio, Harmonics Ratio, Signal Ratio
>Decreasing these setting values allows the Media Channel to detect lesser quality (less clean) in-band DTMF, but it may also result in "false-positives", when some incoming sounds are incorrectly interpreted as DTMF signals.

Minimal Duration
>To be detected as a DTMF signal, an DTMF audio sound should last longer than the specified value. Decreasing this setting value can result in "false-positives", when some incoming sounds are incorrectly interpreted as DTMF signals.

# Codecs

The Media Server supports a set of audio codecs. The following panel allows you to control how these codecs are used:

| Codecs | | | |
|---|---|---|---|
| Name | Priority | | |
| G729 | 5 | 3 | |
| G722 | 5 | 3 | |
| PCMU | 5 | 5 | |
| PCMA | 5 | 4 | |

There are two Priority settings for each codec - the primary and the secondary one.

When a Media Server composes an *initial offer*, the codecs are listed sorted by their primary and secondary priority values.
Codecs with the primary Priority set to Inactive or to 1 are not listed in the initial offer.

When a Media Server composes an *answer* or a non-initial *offer*, i.e. when the audio codecs supported by the peer are known, only the codecs supported by the peer are listed.
Codecs with the primary Priority set to Inactive are not listed in answers and non-initial offers.
The Media Server codecs are listed sorted by their primary Priority values. If several Media Server codecs have the same primary Priority values, they are listed in the same order as they were listed in the peer's offer.

**Note:** the set of supported codecs depends on the platform CommuniGate Pro is running on.

Compressing codecs (such as G.729) require a lot of CPU resources. You may want to offload coding-encoding operations to separate Transcoder servers:

| External Media Servers |
|---|
| Usage:    Disabled |

If the Usage setting is set to `Transcoder`, the Media Server tries to create a *transcoder channel* on one of the listed Transcoder servers. The peer is instructed to exchange media with the created *transcoder channel*, while the *transcoder channel* exchanges media with the CommuniGate Pro Media Server using the PCM (G.711) codec.

Transcoder servers are used in a round-robin manner. If a Transcoder refuses to create a *transcoder channel*, and there are more than 2 servers specified, the next server is contacted.

The CommuniGate Pro Server software or special CommuniGate Pro Transcoder software can be used to implement Transcoder servers.

# Features

The Media Server supports clients connecting from remote NATed networks, using the same algorithms as those employed with the [Media Proxy](#) component. When a call media is terminated with the CommuniGate Pro Media Server, there is no need to build a Media Proxy to handle media transfer over a remote NAT.

The Media Server supports DTMF "codecs": if the peer announces support for such a codec, the Media Server detects audio packets sent using that codec and interprets the packet content as DTMF symbols. The Media Server can also send DTMF symbols using special DTMF "codecs".

If the peer does not announce support for a DTMF "codec", the Media Server detects DTMF symbols sent "in-band", analyzing the audio data received.

# Parlay X Interface

- **Third-Party Call Control**
- **Call Notification**
- **Payment**
- **Account Management**
- **Call Handling**

The CommuniGate Pro Server provides the Parlay X interface its Real-Time, Automated Rules, Billing and other features and functions. The Parlay X is an HTTP/XML protocol. It can used by accessing the CommuniGate Pro HTTP User module, via its `/ParlayX/` realm.

All Parlay X HTTP requests must be authenticated.

The Parlay X interfaces version 2 and 3 are supported.

# Third-Party Call Control

The ParlayX Third-Party Call Control interface allows a client application:

- to initiate a call between 2 parties
- to read the initiated call status
- to cancel the initiated call
- to end the initiated call

When an "initiate call" request is received, CommuniGate Pro starts the `parlayMakeCall` Real-Time application on behalf of the authenticated user.

The application parameters are the request parameters: the calling party and called party addresses, and, optionally, the `charging` parameter.

The `callIdentifier` returned is the Task ID of the started application.

The "cancel call" and "end call" requests are sent to the started application as `cancelCall` and `endCall` events.

The "read status" request returns the contents of the "application status" dictionary set with the started application.

The "add participant" and "delete participant" requests are sent to the started application as `addCallPeer` and `delCallPeer` events; the event parameter contains the participant URI specified in the request.

The "transfer participant" requests are sent to the started application as two events.
First, the `transferTarget` event is sent. Its parameter contains the destination session Task ID.
Then the `transferCallPeer` event is sent. Its parameter contains the participant URI specified in the request.

The application does not quit immediately after the call fails or is terminated. The application continues to run for some period of time (30 seconds by default), processing the "read status" requests for the completed call/session.

# Call Notification

The ParlayX Call Handling interface allows a client application to set Signal Rules for an Account. These Rules include special [Parlay Actions](#) that implement the "CallDirection" and "CallNotification" Parlay Interfaces.

The authenticated users can modify their own Account Signal Rules if they have a proper [Account setting](#) value.
The authenticated users can read and modify Signal Rules for other Accounts if they are granted [Domain Administrator](#) rights.

**Note:** the `stopCallNotification` and `stopCallDirectionNotification` requests must contain the `addresses` and `criteria` parts, while the `correlator` part value is ignored.
This is required since Rules are set individually for each Account, and the `correlator` data does not allow the Server to direct the Rule deletion request to the proper Account.

**Note:** the `handleXxxxResponse` messages can specify the `Fork` action instead of the `Route` action. The address(es) specified with the `routingAddress` part are added to the Signal AOR set, while the current AOR set remains active.

## Payment

The ParlayX Payment interface allows a client application to access the CommuniGate Pro [Billing](#) Manager.

**Note:** CommuniGate Pro Accounts can maintain several Balances. All ParlayX Payment requests can include an `xsd:string`-type `balanceType` element to specify the Account Balance name.

## Account Management

The ParlayX Account Management interface allows a client application to access the CommuniGate Pro [Billing](#) Manager.

# Call Handling

The ParlayX Call Handling interface allows a client application:

- to set Signal Rules for an Account
- to read Signal Rules set for an Account

The authenticated users can modify their own Account Signal Rules if they have a proper [Account setting](#) value.
The authenticated users can read and modify Signal Rules for other Accounts if they are granted [Domain Administrator](#) rights.

All `acceptList` Parlay X elements are converted into one Signal Rule.

All `blockList` Parlay X elements are converted into one Signal Rule.

The `forward` element and each `forwardList` Parlay X elements are converted into three Signal Rules.

**Overview** POP IMAP WebUser XIMSS MAPI AirSync WebDAV FTP TFTP ACAP

# Account Access

- **Access to Accounts**
- **Serving Multiple Domains**
- **Multihoming**
- **Routing**

The CommuniGate Pro Server allows users to use various client applications to access data in their Accounts: Mailboxes, Calendars, Contacts, Files, etc.

- The POP module is a POP3 server; it allows users to retrieve E-mail messages from their INBOX Mailboxes (and, optionally, other mailboxes) using POP3-based mailers.
- The IMAP module is an IMAP4rev1 server; it allows users to process E-mail messages in all Account Mailboxes using IMAP-based mailers.
- The WebMail module is an HTTP (Web) application server; it that allows users to process E-mail messages in all Account Mailboxes and employ other CommuniGate Pro Server features using any Web browser.
- The XIMSS module is an XML Messaging, Scheduling, and Signaling server; it allows users to make and control calls, to process E-mail messages and groupware items in all Account Mailboxes and employ other CommuniGate Pro Server features using "rich" Web-based clients (such as Flash®-based clients).
- The MAPI module is an extension of the IMAP module; it allows users to access their Accounts and Mailboxes via the Microsoft® Windows MAPI (Mail API) and to use the Microsoft Outlook client in its full-featured "groupware" mode.
- The AirSync module is an extension of the HTTP module; it allows users to access their Accounts and Mailboxes via the Microsoft® ActiveSync/AirSync protocol and to use the Windows Mobile clients to receive and send E-mail messages, and to synchronize Contacts, Calendar, and Tasks information with the Server Account data.
- The CalDAV module is an extension of the HTTP module; it allows users to access their Calendar-type and Task-type Mailboxes via the CalDAV protocol.
- The CardDAV module is an extension of the HTTP module; it allows users to access their Contact-type Mailboxes via the CardDAV protocol.
- The HTTP module is an HTTP server; it serves as a transport layer for other modules (such as the AirSync, and WebDAV modules, the XIMSS module in the HTTP Binding mode, etc.), and it provides access to Account File Storage, Mailboxes, and the Account Groupware information.
- The FTP module is an FTP server; it provides access to Account File Storage.
- The TFTP module is a TFTP server; it provides access to Account File Storage.
- The ACAP module is an ACAP server; it allows users to manage their custom application settings stored in their Account datasets.

## Access to Accounts

Every CommuniGate Pro Account can be accessed via the Access modules - POP, IMAP, XIMSS, WebUser Interface, FTP, XMPP, etc. Several client applications can use the same CommuniGate Pro Account at the same time, via the same, or different access modules.

Any Mailbox in any CommuniGate Pro Account can be shared: it can be accessed not only by the Account owner, but also by other users - if the Account owner or an Administrator grants those users access rights for that Mailbox.

# Serving Multiple Domains

The main problem of serving multiple domains on one server is to provide access to accounts in different domains. To look for the specified Account, the server should get the name of the Domain to look in.
Access to Accounts is similar to E-mail delivery and Signal processing: the server needs to know the "full Account name" - an address in the *accountName@domainName* form.

There are several methods to pass the domain name to the server:

- A client application explicitly specifies the domain name.
  - If a user accesses the Server via the HTTP (Web interface), this happens automatically: the user first specifies the server URL (`http://domainname:port`), and then enters the Account name in the Login form.
    Since all modern browsers pass the original URL to the server, the domain name becomes known, and the HTTP module immediately appends that domain name to a simple user name specified in the Login form.
  - If a user accesses the Server using an XMPP, this happens automatically: the user first specifies the server name in the client application settings, and the client application sends that data as the 'to' attribute in the XML streams.
  - If users access the Server with POP or IMAP mailers, they can specify the full account name in the mailer "account name" settings.
    Since many mailers do not accept the `@` symbol in account names, the `%` symbol can be used instead.
    The user `john` with an Account in the secondary Domain client1.com should specify the Account name as `john%client1.com`, not just as `john`.
  - If users access the Server with XIMSS client applications, these applications always specify the full Account name for the authentication operations.

- The domain name can be detected using multihoming. If it is impossible to force users to access the server via the Web interface or to make them enter full account names in their POP/IMAP mailers, multihoming can be used.
  A server is using multihoming if the server computer has more than one Internet (IP) address. Using the Domain Name System (DNS) the secondary domains can be assigned different IP addresses.
  If a secondary Domain has an IP address assigned to it, and a user connects to that IP address, all simple account names specified with the user mailer are processed as names in that Domain.
  It may be difficult to assign an IP address to each Domain, so this method should be used only if it is impossible to make users specify domain names explicitly.

These methods can be used together: a limited number of Domains can be served using dedicated additional IP addresses, while other Domains are served using explicit domain name specifications.

---

# Multihoming

Every access session begins with the authentication procedure: a client application sends a user (Account) name and a password to the Server.

The CommuniGate Pro Server tries to detect which Domain it should use to look for the specified Account name.

- If the specified name contains the `@` symbol or the `%` symbol, the Server assumes that the user has specified a "full account name", i.e. an Account (or other Object) name with its Domain name: `username@domainname` or `username%domainname` (see above).
- If the specified name does not contain the `@` symbol or the `%` symbol, the Server looks at the IP address on which it has received this connection.
  Systems with multihoming (i.e. systems that have several local IP addresses) may have certain IP addresses assigned to some [Domains](#). If a connection IP address is assigned to a some Domain, that domain name is appended to the account name to get the full account name. If the address is dedicated to the Main Domain, the specified name is processed as the Main Domain name.
  By default, all Server IP Addresses are assigned to the Main Domain.

Sample:
> The server computer has 2 IP addresses: `192.0.0.1` and `192.0.0.2`.

The Server Main Domain is `company.com`, and the secondary Domains are `client1.com` and `client2.com`.
The DNS A-records for company.com is pointing to the IP address 192.0.0.1,
the A-record for the client1.com points to a dedicated IP address 192.0.0.2, while the A-records for the client2.com domain point to the same "main" IP address 192.0.0.1.
Each domain has an account `info`.

Three users configure their clients to access an account `info`, but they specify different names in their "server" settings: the first user specifies company.com, the second - client1.com, and the third user specifies client2.com.

When the first user starts her mailer:

- The client application takes the specified "server" setting `company.com`, and it uses the Domain Name System A-records to resolve (convert) that name to the IP address 192.0.0.1.
- The client establishes a connection with that address (which is one of 2 addresses of the Server computer), and it passes the user name `info`.
- The Server detects a simple user name `info` and detects that this connection is established via the Server address 192.0.0.1.
- The Server detects that this IP address is assigned to the Main Domain, so it adds the Main Domain name company.com to the specified simple name.
- The Server gets the correct full account name `info@company.com`.

When the second user starts her client application:

- The client takes the specified "server" setting `client1.com`, and it uses the Domain Name System A-records to resolve (convert) that name to the IP address 192.0.0.2.
- The client establishes a connection with that address (which is one of 2 addresses of the server computer), and it passes the user name `info`.
- The Server detects a simple user name `info` and detects that this connection is established via the server address 192.0.0.2.
- The Server detects that this IP Address is assigned to the client1.com secondary Domain, so it adds that Domain name to the specified simple name.
- The Server gets the correct full account name `info@client1.com`.

When the third user starts her client application:

- The client takes the specified "server" setting `client2.com`, and it uses the Domain Name System A-records to resolve (convert) that name to the IP address 192.0.0.1.
- The client establishes a connection with that address (which is one of 2 addresses of the server computer), and it passes the user name `info`.
- The server detects a simple user name `info` and detects that this connection is established via the Server address 192.0.0.1.
- The Server detects that this IP address is assigned to the Main Domain, so it adds the Main Domain name company.com to the specified simple name.
- The server gets the **incorrect** full account name `info@company.com`.

This happens because the client application (usually - an old POP or IMAP mailer, and FTP client, etc.) has not passed the information about the "server" name from its settings, and the only information the Server had was the IP address.

In order to solve this problem, the third user should specify the account name as `info%client2.com`, not just `info`. In this case, when this user starts the client application:

- The client takes the specified "server" setting `client2.com`, and it uses the Domain Name System A-records to resolve (convert) that name to the IP address 192.0.0.1.
- The client establishes a connection with that address (which is one of 2 addresses of the server computer), and it passes the user name `info%client2.com`.
- The Server detects a full user name `info%client2.com` and it does not look at the IP addresses. It just converts the `%` symbol into the `@` symbol.
- The Server gets the correct full account name `info@client2.com`.

**Note:** most FTP clients work in the same way as the POP/IMAP mailers do, so FTP users are required to supply qualified Account names unless they connect to an IP Address assigned to their Domain.

**Note:** the MAPI Connector always sends a qualified Account Name: if users specify names without the `@` or `%` symbols, the Connector adds the '@' symbol and `Server Name` setting value to the specified account name.

**Note:** the XMPP clients send the 'target domain' name along with the login name. If the specified login name does not contain the `@` or `%` symbols, the Server adds the '@' symbol and "target domain" name to the login name.

# Routing

When the full account name is composed, this name (address) is passed to the [Router](#).

- If the Router reports an error, the client is not authenticated and an error message is returned to the client application. An error is usually the `Unknown user` error, but it can be any error that Router or modules can generate when routing an address.
- If the Router has successfully routed the address, but the address is not routed to the [Local Delivery](#) module, the client is not authenticated and an error message is returned to the client application. This happens if a user has specified a mailing list name, not an account name, or if the specified name is rerouted to some other host (via SMTP, for example).
- If the Router routed the address to some Account in some local Domain, that Account is opened, and the Account passwords are checked.

This means that all routing applied to E-mail and Signal addresses is also applied to the account names specified with mailer applications.

Sample:
    The Account `john` has a `john_smith` alias;
    all E-mail messages and Signals addressed to john_smith will be delivered to the Account john;
    the user can specify either john or john_smith as the "account name" setting in his client applications - in both cases the Account `john` will be opened when those applications connect to the Server.
Sample:
    The Account `john` has been moved from the main domain company.com to the domain client1.com, and it was renamed in j.smith. The administrator has created a Router record:
    `<John> = j.smith@client1.com;`
    all E-mails and Signals addressed to john@company.com will be sent to the Account j.smith in the client1.com Domain;
    the user can still specify just `john` as the "account name" setting, and the same company.com "server" setting in his client application - but the Server will open the Account j.smith in the client1.com Domain.

**Note:**
    do not create any Router record that redirects the Postmaster Account in the Main Domain. You will not be able to administer the Server, unless the postmaster account is redirected to an Account that has the [Master](#) access right.
    If you want the Postmaster E-mails and Signals to be directed to some other user, do not use the Router, but use the postmaster Account [Rules](#) instead.

**CommuniGate Pro**

**Overview POP IMAP WebUser XIMSS MAPI AirSync WebDAV FTP TFTP ACAP**

# POP Module

- **Post Office Protocol (POP3)**
- **Configuring the POP module**
- **User Authentication**
- **Secure (encrypted) Access**
- **Special Features**
- **The XTND XMIT Extension**
- **Notification Alerts**
- **Accessing Additional Mailboxes**
- **Accessing Individual Mail in a Unified Account**

The CommuniGate Pro POP module implements a POP3 server. POP3 servers allow client applications (mailers) to retrieve messages from Account Mailboxes using the POP3 Internet protocol (STD0053, RFC1939, RFC1734, RFC1725) via TCP/IP networks.

The CommuniGate Pro POP module implements several protocol extensions, including the XTND XMIT extension. Some mailers can employ this feature to submit E-mail messages to the CommuniGate Pro Server.

# Post Office Protocol (POP3)

The Post Office Protocol allows computers to retrieve messages from mail server mailboxes. A computer running a mailer (mail client) application connects to the mail server computer and provides account (user) name and the password. If access to the specified user account is granted, the mail application sends protocol commands to the mail server. These protocol commands tell the server to list all messages in the mailbox, to retrieve certain messages, or to delete them. When a server receives a request to retrieve a message, it sends the entire message to the mail client. The mail client may choose to retrieve only the first part of the message.

The POP3 protocol does not provide direct support multi-mailbox Accounts. If a client application specifies a multi-mailbox Account, the INBOX Mailbox is opened.

When the client application sends a request to delete a message from the Mailbox, the message is not deleted immediately, but it is marked by the server. Only when the client application ends the session properly and closes the connection, the marked messages are then removed.

The POP module supports the XTND XMIT extension of the POP protocol. This extension allows users to submit messages via the POP protocol instead of the SMTP protocol.

# Configuring the POP module

Use the WebAdmin Interface to configure the POP module. Open the Access page in the Settings realm:

| Processing | | |
|---|---|---|
| Log Level:  Major & Failures | Channels:  100 | **Listener** |

Use the Log Level setting to specify what kind of information the POP module should put in the Server Log. Usually you should use the `Major` (message transfer reports) or `Problems` (message transfer and non-fatal errors) levels. But when you experience problems with the POP module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.

The POP module records in the System Log are marked with the `POP` tag.

When you specify a non-zero value for the `Maximum Number of Channels` setting, the POP module creates a so-called "listener". The module starts to accept all POP connections that mail clients establish in order to retrieve mail from your server. The setting is used to limit the number of simultaneous connections the POP module can accept. If there are too many incoming connections open, the module will reject new connections, and the mail client should retry later.

By default, the POP module Listener accepts clear text connections on the TCP port 110. The standard TCP port number for secure POP connections is 995, but it is not enabled by default. Follow the listener link to tune the POP Listener.

The POP module supports the STARTTLS command that allows client mailers to establish a connection in the clear text mode and then turn it into a secure connection.

# User Authentication

The POP module allows users to employ all authentication methods supported with the CommuniGate Pro Server, as well as the `APOP` method.

The Account Settings can be used to limit the frequency of POP client logins.

# Secure (encrypted) Access

The POP module can be used to accept SSL/TLS (encrypted) connections from user mailers (see the Listener configuration note above). Additionally, the POP module supports the `STLS` command that allows client mailers to establish plain text, unencrypted connections (using the regular TCP port 110), and then start encrypted communications on those connections.

# Special Features

Unlike many other POP servers, the CommuniGate Pro POP module does not "lock" the Mailbox it opens on a mail clients behalf. The open Mailbox can be used by other client applications at the same time. See the Mailboxes section for the details.

Since the POP3 protocol was not designed to support these features, the CommuniGate Pro POP module:

- shows only the messages that existed in the Mailbox when the Mailbox was opened with the client mailer; all new messages received during this session will be seen by the POP client only when it connects to the Mailbox again;
- keeps *zombies* for the messages deleted during the current session; the module shows them as messages of a zero size, and the module reports an error when a client application tries to retrieve a deleted message;

When a client mailer retrieves a message with the RETR command, the message is marked with the "Seen" flag (this change is noticed when using an IMAP or XIMSS client with the same Mailbox). The TOP command that allows a client POP mailer to retrieve only the first part of the message does not set the Seen flag.

The POP module supports the "empty AUTH" command (the `AUTH` command without parameters), returning the list of supported SASL methods.

# The XTND XMIT Extension

The CommuniGate Pro POP module implements the XTND XMIT protocol extension. Mailer applications that support this extension (like Eudora®) can submit messages to the Server via a POP connection.

This feature can be useful for mobile users that would be otherwise unable to send their messages via CommuniGate Pro SMTP due to the Server anti-spam protection. Submitting messages via POP can be more convenient than using the "address-remembering" scheme, since this method does not have time restrictions.

---

# Notification Alerts

The POP3 protocol does not provide any method to send a notification alert to the client mailer. If an Account has any pending Alert message, the CommuniGate Pro POP Module simply rejects the connection request after the user is authenticated.

The returned error code contains the Alert message text:

```
ALERT: alert message text
```

When the user repeats a connection attempt to the same Account, the next pending Alert message is returned as an error - till all Alert messages are sent to that user.

---

# Accessing Additional Mailboxes

Unlike the IMAP protocol, the POP3 protocol was designed to access only one Account Mailbox - the INBOX Mailbox.

The POP module allows users to access any Account Mailbox by specifying the Mailbox name as a part of the Account name. To access the *mailboxname* Mailbox in the *accountname* Account, a user should specify the account name as: *mailboxname#accountname*:

| Account name (specified in the mailer settings) | Accessed Mailbox |
| --- | --- |
| jsmith | INBOX Mailbox in the jsmith Account |
| private#jsmith | private Mailbox in the jsmith Account |
| lists/info#jsmith@client1.com | lists/info Mailbox in the jsmith Account in the client1.com Domain |

The POP module allows a user to access any Mailbox in any other Account (a *foreign* or *shared* Mailbox), as well as public Mailboxes. See the Mailboxes section for the details.

If a user can log into the *accountname* Account and wants to access the Mailbox *mailboxname* in the *otheraccount* Account, that user should specify the Account name as: *~otheraccount/mailboxname#accountname*:

| Account name (specified in the mailer settings) | Accessed Mailbox |
| --- | --- |
| jsmith | INBOX Mailbox in the jsmith Account |
| ~public/announces#jsmith | the public Mailbox announces |
| ~boss/INBOX#jsmith | INBOX Mailbox in the boss Account |

In all samples above, the user is authenticated as jsmith, using the jsmith account password.

If the authenticated user does not have a right to delete messages in the selected Mailbox, the DELE protocol operations fail and an error code is returned to the user mailer.

The POP module can also use the [Direct Mailbox Addressing](#) feature to open additional Mailboxes.

---

# Accessing Individual Mail in a Unified Account

The POP module implements [Unified Domain-Wide Account](#) filtering. As all [access modules](#), the POP module uses the [Router](#) to process the specified username.

If a client mailer specifies the `abcdef@client1.com` username (as used in the [example](#)), the Router routes this address to the Local account `Cl1`, and it returns `abcdef` as the *local part* of the resulting address.

The POP module checks the local part returned by the Router, and if this string is not empty, it performs filtering on the open Mailbox: the module hides all Mailbox messages that do not have the `X-Real-To` header field (or other field specified in the Local Delivery module settings), or do not have the specified string (individual name) listed in that header field.

So, if the user has specified the `abcdef@client1.com` username, only the messages originally routed to that particular address will be shown in the CL1 Account Mailbox.

If a user connects as `Cl1`, the same Account Mailbox will be opened, but since the local part string will be empty in this case, all Mailbox messages will be shown.

Example:
    The Router line:
    `client1.com = Cl1.local`
    The first message is sent to:
    `abcdef@client1.com`
    It is stored in the Cl1 account INBOX with unique ID 101, and a header field is added:
    `X-Real-To: abcdef`
    The next message is sent to:
    `xyz@client1.com`
    It is stored in the Cl1 account INBOX with unique ID 102, and a header field is added:
    `X-Real-To: xyz`
    After these 2 messages are stored, POP sessions will show different *views* depending on the user name specified:

```
S: +OK CommuniGate Pro POP Server is ready
C: USER Cl1
S: +OK, send pass
C: PASS mypassword
S: +OK 2 message(s)
C: UIDL
S: +OK
S: 1 101
S: 2 102
S: .
C: QUIT
S: +OK bye-bye

S: +OK CommuniGate Pro POP Server is ready
C: USER abcdef@client1.com
S: +OK, send pass
C: PASS mypassword
S: +OK 1 message(s)
C: UIDL
S: +OK
S: 1 101
S: .
C: QUIT
S: +OK bye-bye

S: +OK CommuniGate Pro POP Server is ready
C: USER xyz@client1.com
S: +OK, send pass
C: PASS mypassword
S: +OK 1 message(s)
C: UIDL
S: +OK
S: 1 102
S: .
C: QUIT
S: +OK bye-bye

S: +OK CommuniGate Pro POP Server is ready
C: USER blahblah@client1.com
S: +OK, send pass
C: PASS mypassword
S: +OK 0 message(s)
C: UIDL
S: +OK
S: .
```

```
C: QUIT
S: +OK bye-bye
```

# IMAP Module

- **Internet Message Access Protocol (IMAP)**
- **Configuring the IMAP module**
- **Multi-Access**
- **Access Control Lists**
- **Foreign (Shared) and Public Mailboxes**
- **User Authentication**
- **Non-Mail Mailboxes**
- **Notification Alerts**
- **Login Referrals**
- **Monitoring IMAP Activity**
- **IMAP Implementation Details**
- **Additional IMAP Extensions**

The CommuniGate Pro IMAP module implements an IMAP server. IMAP servers allow client applications (mailers) to retrieve messages from Account Mailboxes using the IMAP4rev1 Internet protocol (RFC2060) via TCP/IP networks.

The IMAP protocol allows client applications to create additional Account Mailboxes, to move messages between Mailboxes, to mark messages in Mailboxes, to search Mailboxes, to retrieve MIME structure of stored messages, and to retrieve individual MIME components of messages stored in Account Mailboxes.

The CommuniGate Pro IMAP module supports both *clear text* and *secure (SSL/TLS)* connections.

# Internet Message Access Protocol (IMAP)

The Internet Message Access Protocol allows client computers to work with messages stored in Mailboxes on remote mail servers. A computer running a mailer (mail client) application connects to the mail server computer and provides account (user) name and the password. If access to the specified user account is granted, the mail application sends protocol commands to the mail server. These protocol commands tell the server to list all messages in the Mailbox, to retrieve certain messages, to delete messages, to search for messages with the certain attributes, to move messages between Mailboxes, etc.

CommuniGate Pro IMAP supports various Internet standards (RFCs) and has many additional unique features.

# Configuring the IMAP module

Use the WebAdmin Interface to configure the IMAP module. Open the Access page in the Settings realm:

| Processing | | |
|---|---|---|
| Log Level: Major & Failures | Channels: 100 | **Listener** |
| | Send 'Running' every: 30 sec | |

Use the Log setting to specify the type of information the IMAP module should put in the Server Log. Usually you should use the `Major` (message transfer reports) or `Problems` (message transfer and non-fatal errors) levels. But when you experience problems with the IMAP module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-

level or link-level details will be recorded in the System Log as well. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.

The IMAP module records in the System Log are marked with the `IMAP` tag.

When you specify a non-zero value for the `Maximum Number of Channels` setting, the IMAP module creates a so-called "Listener". The module starts to accept all IMAP connections that mail clients establish in order to retrieve mail from your server. The setting is used to limit the number of simultaneous connections the IMAP module can accept. If there are too many incoming connections open, the module will reject new connections, and the mail client should retry later.

By default, the IMAP module Listener accepts clear text connections on the TCP port 143, and secure connections - on the TCP port 993. Follow the listener link to tune the IMAP Listener.

The IMAP module supports the STARTTLS command that allows client mailers to establish a connection in the clear text mode and then turn it into a secure connection.

`Send 'Running' every`
> If this setting is not set to `Never`, the IMAP module monitors how long it take to execute the APPEND, COPY, and SEARCH operations. If any of these operations takes longer than the specified period of time, the module sends an "untagged" response to the client application. This feature can be used to prevent client application time-outs, and it can also help in dealing with various NAT boxes that tend to close connections if they show no activity.

---

# Multi-Access

While many other IMAP servers "lock" opened Mailboxes, the CommuniGate Pro IMAP module is designed to provide simultaneous access to any Mailbox for any number of clients.

The IMAP module uses the CommuniGate Pro Mailbox Manager that provides simultaneous access for all types of protocols and clients. See the Mailboxes section for the details.

---

# Access Control Lists

The IMAP module supports RFC2086 (IMAP4 ACL extension). This protocol extension allows IMAP users to grant access to their Mailboxes to other users.

See the Mailboxes section for the detailed description of Mailbox ACLs.

In order to set Access Rights, a client should use a decent IMAP client that supports the ACL protocol extension. If such a client is not available, Mailbox access rights can be set using the WebUser Interface.

---

# Foreign (Shared) and Public Mailboxes

CommuniGate Pro allows account users to access Mailboxes in other Accounts. See the Mailboxes section for the details.

Many popular IMAP clients do not support foreign Mailboxes. There is a workaround for IMAP mailers that use the "subscription" scheme. Subscription is a list of Mailbox names that the mailer keeps on the server. Usually, mailers build the subscription list when you configure them for the first time. Later, they show only the Mailboxes included into the subscription list.

By using a different IMAP client or the WebUser Interface, a user can add a foreign Mailbox name (such as `~sales/processed` or `~public/news/company`) to the subscription list. This will make the old IMAP client show the foreign Mailbox along with the regular account Mailboxes, and the user will be able to work with that foreign Mailbox.

Some IMAP clients do not support foreign Mailboxes at all. To let those clients access shared Mailboxes in other Accounts,

Mailbox Aliases can be used.

# User Authentication

The IMAP module allows users to employ all authentication methods supported with the CommuniGate Pro Server.

If the Domain CLRTXT Login Method option is switched off, and the connection is not encrypted using SSL/TLS, the Server adds the LOGINDISABLED keyword into the list of its supported capabilities.

# Non-Mail Mailboxes

The CommuniGate Pro IMAP module provides access to Mailboxes of all Classes (Calendar, Contacts, etc.). Some clients and/or users can be confused when they see a non-Mail Mailbox.

The module includes non-Mail Mailboxes into its IMAP LIST command response if:

- the Account Non-Mail Mailboxes visible in IMAP setting is enabled, or
- the IMAP LIST command has the CLASS extension, or
- the IMAP ENABLE EXTENSIONS command has been executed.

# Notification Alerts

The CommuniGate Pro IMAP module checks for any pending alert message sent to the authenticated Account. The alert messages are transferred to the client mailer using the standard IMAP [ALERT] response code.

The CommuniGate Pro IMAP module checks for alert messages right after the user is authenticated, and it can detect and send alert messages at any time during an IMAP session.

# Login Referrals

The IMAP module supports RFC2221 (Login Referrals).
As explained in the Access section all user addresses provided with mail clients are processed with the Router.
If the specified user name is routed to an external Internet address (handled with the SMTP module) the IMAP module returns a negative response and provides a login referral. If an IMAP client supports login referrals, it will automatically switch to the new address.

Sample:
A user account j.smith has been moved from your server to the account John at the othercompany.com server. In order to reroute the user mail you have created an alias record in the Router:
`<j.smith> = John@othercompany.com`
Now, when this user tries to connect to his old j.smith account on your server, the server rejects the user name, but provides a login referral:
`1234 NO [REFERRAL IMAP://John;AUTH=*@othercompany.com/] account has been moved to a remote system`
If the mail client supports login referrals, it will automatically try to connect to the server othercompany.com as the user John.

# Monitoring IMAP Activity

You can monitor the IMAP module activity using the WebAdmin Interface. Click the Access link in the Monitors realm to open the IMAP Monitoring page:

| ID | IP Address | Account | Connected | Status | Running |
|---|---|---|---|---|---|
| 9786 | [216.200.213.116] | user1@domain2.dom | 3min | selecting a mailbox | 2sec |
| 9794 | [216.200.213.115] | user2@domain1.dom | 34sec | connected to mailbox | |
| 9803 | [216.200.213.115] | | 2sec | Authenticating | |

ID

This field contains the IMAP numeric session ID. In the CommuniGate Pro Log, this session records are marked with the `IMAP-nnnnn` flag, where *nnnnn* is the session ID.

Address

This field contains the IP address the client has connected from.

Account

This field contains the name of the client Account (after successful authentication).

Connected

This field contains the connection time (time since the client opened this TCP/IP session).

Status

This field contains either the name of the operation in progress or, if there is not pending operation, the current session status (Authenticating, Selected, etc.)

Running

If there is an IMAP operation in progress, this field contains the time since operation started.

If an IMAP connection is used for a [MAPI](#) session, the connection row is displayed with a green background.

IMAP activity can be monitored using the CommuniGate Pro [Statistic Elements](#).

---

# IMAP Implementation Details

The CommuniGate Pro IMAP module implements many IMAP extensions. Some of these extenstions work in the implementation-specific manner.

QUOTA

Each account has its own Quota Root

NAMESPACE

The standard CommuniGate Pro "account name" prefix. The tilda (~) symbol can be used to access Mailboxes in other Accounts.

The `~public` prefix is reported as the Public Namespace prefix, but it is the responsibility of the Domain Administrator to ensure that the Account `public` is created in the proper Domain.

UNSELECT

This IMAP command is equivalent to the `CLOSE` command, but it does not expunge any message marked as `\Deleted`

---

# Additional IMAP Extensions

The CommuniGate Pro IMAP module provides several protocol extensions that are not part of the IMAP standard and are not included into the existing IMAP Extention standards.

COPY

The `ENCRYPTED` *certificateData* parameter can be specified after the target Mailbox name. *certificateData* is either a base64-encoded PKI Certificate, or the asterisk (*) symbol, referring to the personal S/MIME Certificate of the

authenticated user.

The copied messages are [S/MIME-encrypted](#) using the specified certificate.

STATUS

The STATUS command can use the following additional data item names:

INTERNALSIZE

The data item included into the response is a number. This number specifies the size of the Mailbox as it is stored on the server. This size is close to, but not exactly the same as the sum of the RFC822.SIZE attributes of all messages stored in the Mailbox.

OLDEST

The data item included into the response is a date_time string. It specifies the INTERNALDATE of the oldest message stored in the Mailbox. If the Mailbox has no messages, this data item is not included into the response.

UNSEENMEDIA

The data item included into the response is the number of messages that have the Media flag set but do not have the Seen flag set.

Example:
```
A001 STATUS mailbox (UNSEEN OLDEST INTERNALSIZE UNSEENMEDIA)
* STATUS mailbox (UNSEEN 14 OLDEST "23-Feb-2002 07:59:42 +0000" INTERNALSIZE 2345678
UNSEENMEDIA 1)
A001 OK completed
```

LIST

The LIST command can use additional options along with the options specified in the LISTEXT extension standard:

UIDVALIDITY, CLASS, MESSAGES, UIDNEXT, UNSEEN, INTERNALSIZE, OLDEST, UNSEENMEDIA

The data items included into the response have the following format:

\*option_name*(*option_value*)

Example:
```
A001 LIST (CHILDREN CLASS UNSEEN INTERNALSIZE) "" "ma%"
* LIST (\HasNoChildren CLASS("IPF.Contact") \UNSEEN(14) \INTERNALSIZE(2345678) \Unmarked)
mailbox
A001 OK completed
```

APPEND

The APPEND command can use additional options:

REPLACESUID(*number*)

If this extension is specified, after the message is appended, the message with the specified UID (if it exists) is removed from the target Mailbox. Appending and removing is performed as an atomic operation.

CHECKOLDEXISTS

If this extension is specified, it should be specified together with the REPLACESUID extension. When this extension is specified, the message is appended only if the message with the UID specified with the REPLACESUID extension still exists in the target Mailbox. Otherwise an error condition is generated.

Example:
```
A001 APPEND "Drafts" (\Seen \Draft) "20-JAN-2010" (REPLACESUID(30675) CHECKOLDEXISTS) {3450}"
+ Send the message text
```

Additional message flags.

The $MDN, $Hidden, and $Media IMAP message flags are supported, to allow manipulation with these [message flags](#). Adding a $Service message flag to a message effectively removes the message from the "Mailbox view".

ENABLE EXTENSIONS

After this IMAP command has been executed:

- messages with the special $Service message flag become visible
- non-Mail Mailboxes become visible

Overview POP IMAP **WebUser** XIMSS MAPI AirSync WebDAV FTP TFTP ACAP

# WebUser Interface Module

- **WebUser Interface to Multiple Domains**
- **Account Access and WebUser Sessions**
- **Automatic Login and Single Sign-on**
- **WebUser Interface Settings**
- **Configuring Spell Checkers**
- **WebUser Interface to Mailing Lists**
- **Auto Sign-up**

The CommuniGate Pro Server provides Web (HTTP/HTML) access to user Accounts. The WebUser component works via the HTTP module and allows users to read and compose messages and to perform Account and Mailbox management tasks using any Web browser.

Even if a user prefers a regular POP or IMAP mail client, the WebUser Interface can be used to access the features unavailable in some mailers. For example, the WebUser Interface can be used to specify Subscriptions and Access Control Lists for Account Mailboxes - the features many IMAP clients do not support yet. The WebUser Interface can be used to specify Mailbox Aliases, RPOP (External POP) accounts, Account Rules, etc.

This sections describes the WebUser Interface from the administrator point of view. See the WebMail section for more detailed user-level information.

# WebUser Interface to Multiple Domains

When a user points a browser to a WebUser port of the CommuniGate Pro server, the Login page is displayed. The WebUser ports are specified in the HTTP User module settings, the default clear-text WebUser port is 8100, the secure one (not configured by default on some platforms) is 9100.

When the Login page is displayed, users can enter their name and password, and start a WebUser Session.

The WebUser module checks for the domain name specified in the URL and presents the Login page for the addressed Domain. If the CommuniGate Pro server `provider.com` has a secondary Domain `client.com`, then the `<http://provider.com:`*port*`>` URL will display the provider.com Login page in a user browser, and the `<http://client.com:`*port*`>` URL will display the client.com Login page, even if the client.com has no dedicated IP address.

When the WebUser module retrieves the domain name from a URL, it runs it through the Router domain-level records. If the Router Table has a record:

```
www.client.com = client.com
```

the `<http://www.client.com:port>` URL will be processed as the `<http://client.com:port>` URL and it will retrieve the client.com Login page, too.

If the URL specifies a domain name that is not routed to any CommuniGate Pro Domains, an error page is displayed. This usually indicates an error in your Server setup: the specified domain name has a DNS A-record that points to your Server (otherwise the Server would not get this request), but that name is not routed to any of the Server Domains. You should either create a secondary Domain with that name, or route this domain name to one of the existing CommuniGate Pro Domain.

If a URL specifies an IP address instead of a domain name, the WebUser module tries to find a CommuniGate Pro Domain to which the specified address is dedicated. If no Domain is found, the Main Domain Login page is displayed.

Users can open any Account in any Domain from any Login page, if they specify the complete Account name: if the Login page of the Main Domain is displayed (`<http://provider.com:port>`) and the `username@client.com` name is entered in the username field, the Account `username` will be opened in the `client.com` Domain (if the correct password is provided).

If a Domain has some mailing lists, its Login page contain a link to the Mailing List archive pages.

If the Domain has the Auto-Signup option enabled, a link to the Auto Sign-up page is displayed on the Domain Login page.

If the Domain has a custom Security Certificate, a `Certificate` link is displayed. If a user clicks that link, the Domain Certificate can be installed as a *trusted Certificate* in the user browser.

# Account Access and WebUser Sessions

Some protocols (such as IMAP and POP) are session-oriented protocols: a client application establishes a connection with a server, provides the data needed to authenticate the user, processes the data (mailboxes, settings, etc.) in the user account, and then closes the connection. The HTTP protocol is not a session-oriented one: a Web browser establishes a connection, sends one or several page requests, receives the requested data, and closes the connection.

To provide the session-type functionality, the WebUser module implements a so-called *application server*: when a user is authenticated via the "login page", a *virtual session* is created. The virtual session is an internal server data structure keeping the information about the user, open mailboxes, and other session-related data, but it is not linked to any particular network connection. When the user is working with an account using a browser, the WebUser module routes browser requests to one of the already opened virtual sessions.

In order to route requests properly, the WebUser module creates a unique session identifier (session ID) for each virtual session created and makes user browsers include the session ID into every request they send.

To avoid "hijacking" of WebUser sessions, the WebUser module remembers the network (IP) address from which the login request was received, and routes to the session only the requests received from the same IP address.

**Note:** Sometimes, when a user connects via a proxy server, the user requests may come to the Server from different IP addresses (if the proxy server uses several network addresses). In this case, the user should disable the address-controlling option on the WebUser Interface Settings page. Users of some large providers access the Internet via the provider's proxy servers, so their accounts should have the address-controlling option disabled.
Alternatively, enter the provider proxy IP addresses as a range into the NAT Server IP Addresses list.
All network IP addresses that belong to the same range in that list are treated as "same".

To avoid "hijacking" of WebUser sessions, the WebUser module can use HTTP "cookies". When the Use Cookies option is enabled, the Server generates some random "cookie" string and sends it to the user browser when a session is initiated. The browser then always sends that string back to the Server when it tries to access any of the session pages. The Server allows the user to access the session data only when the valid "cookie" string is sent.

**Note:** Some browsers do not support "cookies" or can be configured not to support them. The user should check the browser options before enabling the Use Cookies option.

Usually, users start WebUser sessions by entering their Account names and passwords into the WebUser Interface login page fields. This is a "clear text" login method, and it is secure only when the page is accessed via secure (SSL/TLS) connection (via the `https://` URL).

Alternatively, users can retrieve the `/login/` URL on your Server. The Server will require an HTTP-level Authentication, and the browser will either present the Authentication dialog box, or it will send the user's Certificate if a secure (SSL/TLS) connection is used.

# Automatic Login and Single Sign-on

When designing a set of Web-based services ("portals"), it may be necessary to create a WebUser session without presenting a login page to the user. The following mechanisms can be used.

Automating the Login page
> direct the user browser to
> http://*your.server.domain*[:*port*]/?username=*accountName*&password=*password*

HTTP Authentication
> direct the user browser to
> http://*your.server.domain*[:*port*]/login/
> If the browser automatically logs into this realm, it will not present the Login dialog to the user.

Certificate Authentication
> direct the user browser to
> http://*your.server.domain*[:*port*]/login/
> If the Domain supports Client Certificates, and the proper certificate is installed on the user's computer, the browser automatically logs into this realm, without presenting the Login dialog to the user.

Creating a Session via CLI
> use the Network CLI/API to create new WebUser session, then direct the user browser to
> http://*your.server.domain*[:*port*]/Session/*sessionID*/Hello.wssp

Automating the Login page with other Session ID
> direct the user browser to
> http://*your.server.domain*[:*port*]/?username=*accountName*&password=*sessionID*&SessionIDMethod=yes
> This is the same mechanism as the automated Login page mechanism, but the SessionID Authentication method is used instead of the clear-text Authentication method. The *sessionID* is a the ID of any existing WebUser or XIMSS session with the same *accountName* Account.

---

# WebUser Interface Settings

To configure the WebUser Interface module, use any Web browser to connect to the CommuniGate Pro WebAdmin Interface, open the Access pages in the Settings realm, and then open the Sessions page.

| User Sessions | | | |
|---|---|---|---|
| Log Level: | Major & Failures | Limit: | 3000 |
| Inactivity Time-Out: | 30 minutes | Session Time Limit: | 6 hour(s) |
| Compose Limit: | | Recipients: | 50 |

Limit
> Use this setting to specify the maximum number of concurrent WebUser Interface sessions.
> **Note:** remember that browser (HTTP) connections are not the same as WebUser sessions. It is usually enough to support 100 concurrent HTTP channels to serve 5000 Sessions.

Log
> Use this setting to specify what kind of information the WebUser Interface module should put in the Server Log. Usually you should use the Major (message transfer reports) level.
> The WebUser Interface module records in the System Log are marked with the WEB tag.

Inactivity Time-Out
> Use this setting to specify the maximum time interval between client (browser) connections for a particular WebUser Session. This setting allows to disconnect the users who did not log out correctly, but simply closed their browsers or moved to a different Web site. Do not set this setting to a too small value, otherwise users can get disconnected while they are composing letters.

`Session Time Limit`
Use this setting to specify the maximum "login" time for a WebUser session. The limit is checked when a browser connects and retrieves a page from the session, so it is useless to set this setting to a value that is smaller than the Inactivity Time-Out setting value.

`Compose Limit: Recipients`
Use this setting to specify the maximum number of E-mail addresses in messages composed using the WebUser Interface.

Open the HTTP User Module settings, and find the Sub-Protocols panel:

| Sub-Protocols |
|---|
| **Access** |
| WebUser:   clients |

The Access setting specifies who can create WebUser sessions.

---

# Configuring Spell Checkers

The Server Administrator can specify one or several external spell checker programs. To configure spell checkers, follow the Spelling link on the General Settings page.

The Spelling page appears :

| Enabled | Language | Log Level | Program Name and Parameters |
|---|---|---|---|
| | | Major & Failures | |
| | | Convert to: | Western European (ISO) |
| | | Major & Failures | |
| | | Convert to: | Western European (ISO) |
| | | Major & Failures | |
| | | Convert to: | Cyrillic (KOI) |
| | | Major & Failures | |
| | | Convert to: | Western European (ISO) |

To configure a spell checker, specify the language the program can process, and path to the program file, and the character set the program can handle. The internal data presentation use the UTF-8 character set, so set the `UTF-8` value if no conversion is needed.

Use the Log Level setting to specify the type of information the spell checker module should put in the Server Log.

The spell checker module records in the System Log are marked with the `SPELLER` tag.

Use the Enable check box to enable and disable the spell checker program without removing it from the program list.

To remove a spell checker program, enter an empty string into its Language field and click the Update button.

The spell checker program should provide the same "pipe" interface as the popular `Ispell` and `aspell` programs:

- Text is sent to the program line-by-line with the first symbol of the line being a space symbol;
- For each input line, the program returns zero, one, or several non-empty response lines followed by an empty line.
- Only the response lines that start with the ampersand (`&`) or hash (`#`) symbols are processed.
- The hash-line has the following format:
  `# original offset`

where *original* is a misspelled word, and *offset* is the position of that word in the input text line.

- The ampersand-line has the following format:
  ```
  & original count offset:suggestion1, suggestion2, ...
  ```
  where *original* is a misspelled word, and *offset* is the position of that word in the input text line, *count* is the number of *suggestions*.

# WebUser Interface to Mailing Lists

The WebUser module presents a link to the Mailing Lists page on the Domain Login Page.

The Mailing Lists page displays all mailing lists created in the Domain that have the `allow anybody to browse` option enabled. Each name is a link that can be used to open a page listing messages in the Mailing List archive. Since Mailing Lists messages are archived in Mailboxes, the Mailing List WebUser interface is similar to the Mailbox Browsing interface.

The Mailing List WebUser Interface does not require any authentication, so no virtual session is created for list users, and each browser request is processed independently.

# Auto Sign-up

If a domain has the Auto-Signup option enabled, the WebUser Interface Login page contains a link to the Auto-Signup page. This page allows a new user to enter a user name, a password, the "real-life" name, and to create a new Account.

When a new account is created, its options and settings are taken from the domain Account Template.

| Introduction |
| --- |
| Installation |
| SysAdmin |
| Network |
| Objects |
| Storage |
| E-mail |
| Real-Time |
| Access |
| Services |
| Directory |
| Clusters |
| Applications |
| Miscellaneous |
| Licensing |
| |
| WebMail |
| Pronto! |
| PBX |

**Overview** **POP** **IMAP** **WebUser** **XIMSS** **MAPI** **AirSync** **WebDAV** **FTP** **TFTP** **ACAP**

# XIMSS Module

- **Configuring the XIMSS Module**
- **XIMSS Connections to Other Modules**
- **Flash Security**
- **XIMSS Sessions**
- **HTTP Binding**
    - **HTTP Login**
    - **HTTP Synchronous Communications**
    - **HTTP Asynchronous Communications**
- **Monitoring XIMSS Activity**

The CommuniGate Pro XIMSS module implements an open XIMSS protocol. This protocol supports advanced clients allowing them to access all Server functions in a secure and effective way.

The CommuniGate Pro XIMSS module supports both *clear text* and *secure (SSL/TLS)* connections.

The the XIMSS Protocol section for more details on the protocol and its features.

The CommuniGate Pro XIMSS module requires either a Groupware-type License Key or a special XIMSS License Key. The Server accepts up to 5 concurrent XIMSS sessions without these Keys.

# Configuring the XIMSS Module

Use the WebAdmin Interface to configure the XIMSS module. Open the Access page in the Settings realm:

| Processing | | |
| --- | --- | --- |
| Log Level: Major & Failures | Channels: 100 | **Listener** |

Use the Log setting to specify the type of information the XIMSS module should put in the Server Log. Usually you should use the `Major` (message transfer reports) or `Problems` (message transfer and non-fatal errors) levels. But when you experience problems with the XIMSS module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well. When the problem is solved, set the Log Level setting to its regular value, otherwise your System Log files will grow in size very quickly.

The XIMSS module records in the System Log are marked with the `XIMSSI` tag.

When you specify a non-zero value for the `Maximum Number of Channels` setting, the XIMSS module creates a Listener. The module starts to accept all XIMSS connections that clients establish in order to communicate with your Server. The setting is used to limit the number of simultaneous connections the XIMSS module can accept. If there are too many incoming connections open, the module will reject new connections, and the client should retry later.

By default, the XIMSS module Listener accepts clear text connections on the TCP port 11024. Follow the Listener link to tune the XIMSS Listener.

# XIMSS Connections to Other Modules

XIMSS connections can be made to TCP ports served with other CommuniGate Pro modules. If the first symbol received on a connection made to the [HTTP](#) module is the < symbol, the HTTP module passes the connection to the XIMSS module.

When a connection is passed:

- the logical job of the passing module completes.
- the logical job of the XIMSS module is created, in the same way when an XIMSS connection is received on a port served with the XIMSS module.
- the XIMSS module restrictions for the total number of XIMSS channels and for the number of channels opened from the same IP address are applied.

When all users initiate XIMSS connections via other Module ports, you can disable the XIMSS [Listener](#) by setting all its ports to zero.

# Flash Security

When a Flash client connects to an *XMLSocket* server (such as the CommuniGate Pro XIMSS module), it can send a special `policy-file-request` request. The XIMSS module replies with an XML document allowing the client to access any port on the Server.

# XIMSS Sessions

When a user is authenticated, the XIMSS module creates a XIMSS session. The current XIMSS module TCP connection can be used to communicate with that session.

A XIMSS session can be created without the XIMSS module, using special requests sent to the [HTTP](#) User module. See the [XIMSS Protocol](#) section for more details.

The XIMSS session records in the System Log are marked with the `XIMSS` tag.

# HTTP Binding

A client application can access the XIMSS interface via HTTP connections.

A client application should start by sending an HTTP Login request to create a new XIMSS session.

When a XIMSS session is created, the client application can send XIMSS protocol requests to it and receive XIMSS protocol responses from the session using HTTP requests.

Client applications can use `GET` and `POST` HTTP requests.
If a request contains a body, it is assumed to be an XML text, unrelated to the actual value of the Content-Type header field. The XML text must be a `<XIMSS/>` element.
If a request produces a non-empty response body, the body is always an XML text containing one `<XIMSS/>` element, and the response Content-Type header field is `text/xml`.

Open the HTTP User Module settings, and find the Sub-Protocols panel:

| Sub-Protocols |
|---|
| **Access** |
| XIMSS:   clients |

The Access setting specifies who can create XIMSS sessions using HTTP Binding.

## HTTP Login

To start a XIMSS session, a client application should send an HTTP request to the CommuniGate Pro HTTPU module using the following URLs:

```
        http://domainName[:port]/ximsslogin/
    or
        https://domainName[:port]/ximsslogin/
```

If the request contains the `userName` parameter, the Server tries to authenticate the specified user (Account):

- If the `password` parameter is present, the regular clear-text method is used.
- If the `nonce` parameter is present, the [CRAM-MD5](#) method is used. The "nonce" parameter value should be a value received as part of a `features` response (see below), it should be a valid "authentication nonce". The request must contain the `authData` parameter containing the base64-encoded CRAM-MD5 "challenge response".
- If the `sessionid` parameter is present, the [SessionID](#) method is used.
- If the `errorAsXML` parameter is present and the login operation fails, the error condition is returned not as an HTTP result code with an HTML error page, but as an `<response/>` element with `errorNum` and `errorText` attributes, enclosed into a `<XIMSS/>` element.
- If the `version` parameter is present, its value specifies the protocol version the client implements (see the [Login](#) operation parameters).

If the `userName` parameter is absent, the Server tries to authenticate the request using the [TLS Client Certificate](#) (if specified), or using the HTTP authentication methods.
This functionality is the same as the WebUser Interface [Automatic Login and Single Sign-on](#) functionality, but the `/ximsslogin/` URL is used.

A request to the `/ximsslogin/` URL can contain a `text/xml` body. In this case, no login operation is performed.
The XML body should contain one `<XIMSS>` element containing zero, one, or several XIMSS [Pre-Login operations](#). The Server sends an HTTP response with XML data. The response is a `<XIMSS>` element containing the requested operations result.

Example:
```
C: GET /ximsslogin/ HTTP/1.1
   Host: myserver.com
   Content-Type: text/xml
   Content-Length: 42

   <XIMSS><listFeatures id="list" /><XIMSS>

S: HTTP/1.1 200 OK
   Content-Length: 231
   Connection: keep-alive
   Content-Type: text/xml;charset=utf-8
   Server: CommuniGatePro/5.3

   <XIMSS><<features id="s" domain="x.domain.dom"><starttls/><sasl>LOGIN</sasl><sasl>PLAIN</sasl><sasl>CRAM-
MD5</sasl><sasl>DIGEST-
MD5</sasl><sasl>GSSAPI</sasl><nonce>2C3E575E5498CE63574D40F18D00C873</nonce><language>german</language><signup/></features><response
id="s"/></XIMSS>
```

If the user has been successfully authenticated, and the XIMSS session has been created, the HTTP Login response contains the XIMSS [session message](#) with the session ID string. Note that the `session` message does not contain the `id` attribute.

Example:
```
C: GET /ximsslogin/?userName=account@domain&password=abcd&version=6.1 HTTP/1.1
   Host: myserver.com
   Content-Length: 0

S: HTTP/1.1 200 OK
   Content-Length: 105
   Connection: keep-alive
   Content-Type: text/xml;charset=utf-8
   Server: CommuniGatePro/5.3

   <XIMSS><session urlID="562-kAI2lxNBR4ApmHg4wiW9" userName="account@domain" realName="J. Smith" version="6.1.2" /></XIMSS>
```

Alternative URLs can be used to start a XIMSS session using the [TLS Client Certificate](#), or using the HTTP authentication methods:

```
        http://domainName[:port]/auth/ximsslogin/
    or
        https://domainName[:port]/auth/ximsslogin/
```

This method is useful if an application first retrieves an HTML page or some other document using the [`/auth/` realm](#), forcing the browser to ask the user for credentials, and then the application creates a XIMSS session for the same user, as the browser will resend the same credentials when sending a request to the `/auth/ximsslogin/` URL.

## HTTP Synchronous Communications

A client should send requests to a created XIMSS session use the following Session URL:

> `http://`*`domainName`*`[:`*`port`*`]/Session/`*`sessionID`*`/sync`
> or
> `https://`*`domainName`*`[:`*`port`*`]/Session/`*`sessionID`*`/sync`

where *sessionID* is the `session` message urlID attribute.

The HTTP request body should contain one <XIMSS /> element, with zero, one, or more XIMSS protocol requests.

The Server returns one <XIMSS /> element in the HTTP response body. This element contains the XIMSS protocol `response` messages (one for each XIMSS request sent, in the same order), and all synchronous data messages generated with the submitted XIMSS requests.

Example:
```
C:POST /Session/562-kAI2lxNBR4ApmHg4wiW9/sync HTTP/1.1
  Host: myserver.com
  Content-Length: nnn

  <XIMSS><noop id="i1" /><readTime id="i2" /></XIMSS>

S:HTTP/1.1 200 OK
  Content-Length: nnn
  Connection: keep-alive
  Content-Type: text/xml;charset=utf-8
  Server: CommuniGatePro/5.3

  <XIMSS><response id="i1"/><currentTime id="i2" gmtTime="20070502T083313" localTime="20070502T003313"/><response
id="i2"/></XIMSS>
```

If a XIMSS client works in an unreliable environment, where it may have to resend HTTP requests, then each non-empty HTTP request should contain a `reqSeq` parameter. This parameter value should be increased by 1 for each new HTTP request sent.
If the Server receives an HTTP request with the same `reqSeq` parameter as the previously received and processed HTTP request, then the Server resends the last response (one it has sent to the previous HTTP request wit the same `reqSeq`).
If the Server receives an HTTP request with the `reqSeq` parameter not equal to the `reqSeq` parameter of the previously received request and not equal to the `reqSeq` parameter of the previously received request increased by 1, then the Server returns an error.

A client application can use an "empty request" (an HTTP request without a body) to read asynchronous XIMSS data messages.

When such an empty request is received, the Server checks if there is any pending asynchronous data messages for the specified session. If there is no pending asynchronous data messages, the request is held until either:

- an asynchronous data message is generated for the session; or
- the waiting time is over; or
- a new "empty request" is received; or
- the session is closed.

An empty request can specify the waiting time as the `maxWait` parameter (number of seconds).

If no data messages were retrieved, the Server sends a response containing an empty `<XIMSS/>` element, without any attributes.

If some data messages were retrieved, the Server sends a response (an "asynchronous response") containing one `<XIMSS/>` element, with the `respSeq` attribute. This attribute contains the sequence number for this `<XIMSS/>` response element.

For each session, the Server keeps the last "asynchronous response" composed.

Each empty request should contain a `ackSeq` parameter. It should contain the `respSeq` value of the last received asynchronous response.
If the client has not received any asynchronous response yet, this parameter value must be `0`.

When the Server receives an empty request with the `ackSeq` equal to the `respSeq` value of the kept last composed asynchronous response, it considers that response as "acknowledged", and removes it.

When the Server receives an empty request with the `ackSeq` equal to the `respSeq` value of the last composed asynchronous response minus one (respSeq-1), and it still keeps this composed response, the Server resends that response to the client. As a result, if the client encounters any communication error while doing an "empty request" HTTP transaction, it can resend that empty request.

An empty request without an `ackSeq` parameter acknowledges all "asynchronous responses" composed and kept.

When a server returns an empty `<XIMSS/>` element, the next empty request can contain either no `ackSeq` parameter, or the same `ackSeq` parameter as the previous empty request. Because of this subsequent empty requests may use the same request URL and the same parameters, and the client platform may return the previous cached `<XIMSS/>` element result immediately, without sending the request to the server.

To avoid this problem, include the `getSeq` parameter into each empty request, increasing its value after a successful transaction.

Example:

```
C:GET /Session/562-kAI2lxNBR4ApmHg4wiW9/get?maxWait=90&ackSeq=0&getSeq=0 HTTP/1.1
   Host: myserver.com
   Content-Length: 0

...optional pause (up to 90 seconds)...
S:HTTP/1.1 200 OK
   Content-Length: 10
   Connection: keep-alive
   Content-Type: text/xml;charset=utf-8
   Server: CommuniGatePro/5.3

   <XIMSS/>
C:GET /Session/562-kAI2lxNBR4ApmHg4wiW9/get?maxWait=90&ackSeq=0&getSeq=1 HTTP/1.1
   Host: myserver.com
   Content-Length: 0

...optional pause (up to 90 seconds)...
S:HTTP/1.1 200 OK
   Content-Length: nnn
   Connection: keep-alive
   Content-Type: text/xml;charset=utf-8
   Server: CommuniGatePro/5.3

   <XIMSS respSeq="1"><folderReport folder="INBOX" mode="notify" /></XIMSS>
response did not reach the client, client is resending the request
C:GET /Session/562-kAI2lxNBR4ApmHg4wiW9/get?maxWait=90&ackSeq=0&getSeq=1 HTTP/1.1
   Host: myserver.com
   Content-Length: 0

S:HTTP/1.1 200 OK
   Content-Length: nnn
   Connection: keep-alive
   Content-Type: text/xml;charset=utf-8
   Server: CommuniGatePro/5.3

   <XIMSS respSeq="1"><folderReport folder="INBOX" mode="notify" /></XIMSS>
C:GET /Session/562-kAI2lxNBR4ApmHg4wiW9/get?maxWait=90&ackSeq=1&getSeq=2 HTTP/1.1
   Host: myserver.com
   Content-Length: 0

...optional pause (up to 90 seconds)...
S:HTTP/1.1 200 OK
   Content-Length: 10
   Connection: keep-alive
   Content-Type: text/xml;charset=utf-8
   Server: CommuniGatePro/5.3

   <XIMSS/>
```

# HTTP Asynchronous Communications

A client can send requests to a created XIMSS session so that all responses (including the `response` messages and synchronous data messages) are returned only in response to the "empty requests".

> http://*domainName*[:*port*]/Session/*sessionID*/async
> or
> https://*domainName*[:*port*]/Session/*sessionID*/async

where *sessionID* is the `session` message urlID attribute.

The HTTP request body should contain one <XIMSS /> element, with zero, one, or more XIMSS protocol requests.

All generated `response` messages (one for each XIMSS request sent, in the same order), and all synchronous data messages generated with the submitted XIMSS requests are re-submitted to the XIMSS session as asynchronous messages. The Server returns an empty HTTP response.

Example (single connection, polling):

```
C:GET /Session/562-kAI2lxNBR4ApmHg4wiW9/get?maxWait=0&ackSeq=0&getSeq=0 HTTP/1.1
   Host: myserver.com
   Content-Length: 0

S:HTTP/1.1 200 OK
   Content-Length: 10
   Connection: keep-alive
   Content-Type: text/xml;charset=utf-8
   Server: CommuniGatePro/5.3
```

```
   <XIMSS/>
C:POST /Session/562-kAI2lxNBR4ApmHg4wiW9/async HTTP/1.1
   Host: myserver.com
   Content-Length: nnn

   <XIMSS><noop id="i1" /><readTime id="i2" /></XIMSS>
S:HTTP/1.1 200 OK
   Content-Length: 0
   Connection: keep-alive
   Content-Type: text/plain;charset=utf-8
   Server: CommuniGatePro/5.3

C:GET /Session/562-kAI2lxNBR4ApmHg4wiW9/get?maxWait=0&ackSeq=0&getSeq=1 HTTP/1.1
   Host: myserver.com
   Content-Length: 0

S:HTTP/1.1 200 OK
   Content-Length: nnn
   Connection: keep-alive
   Content-Type: text/xml;charset=utf-8
   Server: CommuniGatePro/5.3

   <XIMSS respSeq="1"><response id="i1"/><currentTime id="i2" gmtTime="20070502T083313" localTime="20070502T003313"/><response
id="i2"/></XIMSS>
```

Example (2 connections, waiting):

```
C:GET /Session/562-kAI2lxNBR4ApmHg4wiW9/get?ackSeq=0&getSeq=0 HTTP/1.1
   Host: myserver.com
   Content-Length: 0

...waiting...
                                                           C:POST /Session/562-kAI2lxNBR4ApmHg4wiW9/async HTTP/1.1
                                                              Host: myserver.com
                                                              Content-Length: nnn

                                                              <XIMSS><noop id="i1" /><readTime id="i2" /></XIMSS>
S:HTTP/1.1 200 OK                                          S:HTTP/1.1 200 OK
   Content-Length: nnn                                        Content-Length: 0
   Connection: keep-alive                                     Connection: keep-alive
   Content-Type: text/xml;charset=utf-8                       Content-Type: text/xml;charset=utf-8
   Server: CommuniGatePro/5.3                                 Server: CommuniGatePro/5.3

   <XIMSS respSeq="1">
     <response id="i1"/>
     <currentTime id="i2" gmtTime="20070502T083313"
       localTime="20070502T003313"/>
     <response id="i2"/>
   </XIMSS>
C:GET /Session/562-kAI2lxNBR4ApmHg4wiW9/get?ackSeq=1&getSeq=1 HTTP/1.1
   Host: myserver.com
   Content-Length: 0

...waiting...
```

# Monitoring XIMSS Activity

You can monitor the XIMSS Module activity using the WebAdmin Interface.

Click the Access link in the Monitors realm to open the Access Monitoring page:

| | | | | | 3 of 3 selected |
|---|---|---|---|---|---|
| **ID** | **IP Address** | **Account** | **Connected** | **Status** | **Running** |
| 9786 | [216.200.213.116] | user1@domain2.dom | 3min | listing messages | 2sec |
| 9794 | [216.200.213.115] | user2@domain1.dom | 34sec | reading request | |
| 9803 | [216.200.213.115] | | 2sec | authenticating | |

ID
> This field contains the XIMSS numeric session ID. In the CommuniGate Pro Log, this session records are marked with the `XIMSS-nnnnn` flag, where *nnnnn* is the session ID.

IP Address
> This field contains the IP address the client has connected from.

Account
> This field contains the name of the client Account (after successful authentication).

Connected
> This field contains the connection time (time since the client opened this TCP/IP session).

Status
> This field contains either the name of the operation in progress or, if there is not pending operation, the current session status (Authenticating, Selected, etc.).

`Running`
     If there is an XIMSS operation in progress, this field contains the time since operation started.


XIMSS activity can be monitored with the CommuniGate Pro [Statistic Elements](#).

# MAPI Connector

- **MAPI Connector Overview**
- **Installing the MAPI Connector**
- **Creating a Mail Profile**
- **Configuring the MAPI Connector**
  - **Server**
  - **Account Settings**
  - **Connection**
  - **Advanced**
- **Enabling Mailbox Sharing**
- **Delegation**
- **Free/Busy Information**
  - **Posting Free/Busy Information**
  - **Accessing Free/Busy Information for Other Users**
- **Working Offline**
- **Configuring Automatic Rules**
- **WebMail Integration**
- **Communicating with Microsoft Exchange users**
- **Real-Time Communications**
- **Server-side Encryption**
- **Troubleshooting**
- **Known Limitations**

The CommuniGate Pro Server can be used as a "service provider" for Microsoft Windows applications supporting the MAPI (Microsoft Messaging API). To use this service, a special Connector library (CommuniGate Pro MAPI Connector `dll`) should be installed on client Microsoft Windows workstations.

The CommuniGate Pro MAPI Connector requires either a Groupware-type License Key or a special MAPI License Key.

You can review the CommuniGate Pro MAPI Connector [Revision History](#).

## MAPI Connector Overview

MAPI stands for **M**essaging **A**pplication **P**rogramming **I**nterface, the system component that the Microsoft corporation has included into its Windows® operating system and the API to use that component with Windows applications.

The MAPI infrastructure provides an additional level of abstraction. Windows applications do not deal directly with a groupware server (or any other "data store"). Instead, applications send Messaging requests (such as "list my mailboxes", "retrieve message number X", etc.) to the MAPI component, and the MAPI component uses the installed "Connector" modules to send those requests to an Exchange® server, to locally stored "personal folders", to a fax server, etc.

The expandable nature of the MAPI architecture allows for creation of additional "Connectors" that can interact with various server products. One of the problems that such a Connector has to solve is data format: Windows applications send data objects via MAPI to Connector modules in the so-called "MAPI object" format that has very little in common with any Internet format. The CommuniGate Pro MAPI Connector converts the MAPI data into one of the standard Internet formats and stores the converted "messaging objects" as standard Internet messages in a CommuniGate Pro Mailbox. When reading those Mailboxes, the CommuniGate Pro MAPI Connector converts messages back into the "MAPI object" format and passes the converted

objects back to MAPI and Windows applications (such as Outlook).

The CommuniGate Pro MAPI Connector acts as a "MAPI provider". It accepts Messaging API requests from Microsoft Outlook (Outlook 2002, Outlook XP and later) running in the "groupware" mode, and from other Windows applications. The MAPI Connector converts these requests into extended IMAP commands and sends them to the CommuniGate Pro Server.

The CommuniGate Pro MAPI Connector also performs data conversion between proprietary Microsoft "objects" data formats and the standard Internet data formats.
Because the standard Internet formats are used, messages stored with the CommuniGate Pro MAPI Connector can be read using any standard POP3 or IMAP mail client, the CommuniGate Pro WebUser Interface, or any XIMSS-based client.

The CommuniGate Pro MAPI Connector uses TCP/IP networks and should be configured to connect to any non-TLS (clear text) IMAP port of your CommuniGate Pro server (the port 143 is the standard IMAP port).

The CommuniGate Pro MAPI Connector supports both *clear text* and *secure* (SSL/STARTTLS) connections, and it can use plain text and secure CRAM-MD5 login methods.

The CommuniGate Pro MAPI Connector contains two code parts (shared libraries). The *starter code* part should be installed on Windows workstations. It provides the configuration interface and it is used to connect to the CommuniGate Pro server. The main MAPI Connector functionality is implemented as a shared library stored in the Server application directory, and it is called the *server code* part.
When the MAPI Connector *starter code* connects to the CommuniGate Pro Server, the Server sends the *server code* part of the MAPI Connector to the client computer.
This method allows you to deploy "regular" MAPI Connector updates by updating your CommuniGate Pro Server software only, without running the MAPI Connector Installer on all client workstations.
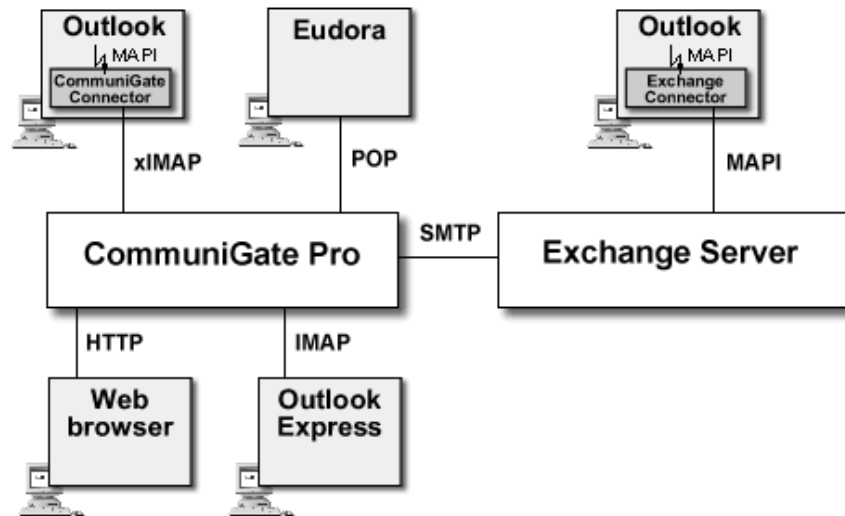


# Installing the MAPI Connector

You need to install the MAPI Connector *starter* code part shared library (a `.dll` file) on Microsoft Windows workstations. Download the MAPI Connector archive and unpack it. The unpacked folder contains the `Setup.exe` file.

Start the unpacked `Setup.exe` application to install or update your CommuniGate Pro MAPI Connector software. After successful install, the application may ask you to re-create your Mail Profile.

You can use the same `Setup.exe` application to uninstall the MAPI Connector software from the workstations.

The MAPI Connector `Setup.exe` program can be started in the *silent* mode (without any user interface dialogs) using the following command-line parameters:

`/i`          install or upgrade the MAPI connector

`/r`          remove the MAPI connector

| `/q` | do not ask for profile configuration |
| `/Q` | do not display error messages |

You may want to use the silent mode to install or upgrade the MAPI Connector *starter code part* on many workstation, using the Windows network administration methods and tools.

The `cgmxui32.inf` file can be used to automatically set the the MAPI Connector Account Settings and Shared Accounts. Download this file and modify it to fit your needs, then place it into the same directory where the `Setup.exe` application resides. The application will use it during the installation process.

---

# Creating a Mail Profile

When the CommuniGate Pro MAPI Connector is installed on a client workstation, you can create a Mail Profile that will tell Outlook and other applications to use the CommuniGate Pro MAPI services.

If you use Outlook 98 or Outlook 2000 check that it is configured to run in the "groupware mode". Start Outlook, and select the `Options` item from the `Tools` menu. The Options dialog box appears. Select the `Mail Services` Tab and click the `Reconfigure Mail Support` button to open the `E-mail Service Options` dialog box. Check that the `Corporate or Workgroup` option is selected.
**Note**: you may need the MS Office installation CD-ROM to complete Outlook mode switching.
**Note**: the latest versions of the MAPI Connector (1.54 and up) rely on the Unicode-enabled interfaces of MAPI that are not compatible with Outlook 98 and Outlook 2000.

Open the `Mail` Control Panel and click the `Show Profiles` button. The list of Mail profiles appears. If the CommuniGate Pro MAPI Installer has instructed you to re-create your existing Profile, select the old Profile and click the `Remove` button.

Click the `Add` button to create a new Profile. Depending on the version of the Outlook and the Mail control panel installed, you will see several dialog boxes. If you see a dialog box with the `Additional Server Types` option, select that option. Select `CommuniGate Pro Server` as the "service" or "additional server type".

You can add other services into the same Profile.

---

# Configuring the MAPI Connector

When the CommuniGate Pro service is added to a Mail Profile, the service settings can be configured. Later you can open the `Mail` control panel, open this Profile, and open the `CommuniGate Pro Server` settings. You can also use the `Services` item in the Outlook `Tools` menu to open the service settings.

## Server

The Server panel allows you to specify the CommuniGate Pro Server and Account data:
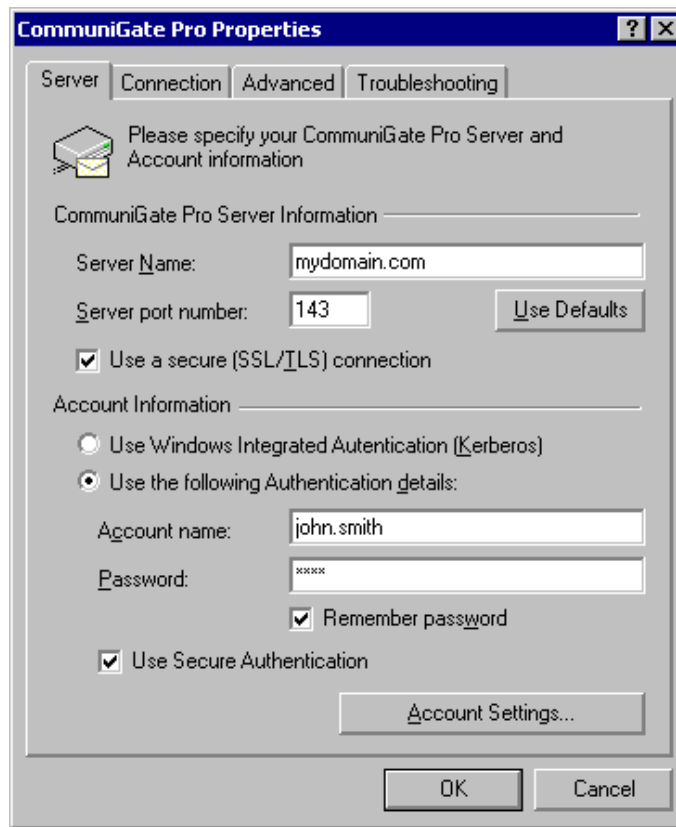
**Server Name**
> The name of your CommuniGate Pro Server. This should be a domain (DNS) name that has an A-record pointing to the network (IP) address of the server.
> **Note:** the MAPI Connector adds this name to the Account Name (see below) to send fully-qualified account names to the Server. This feature simplifies multi-domain support using a single IP address. Make sure that the specified name is either a name of some CommuniGate Pro Domain, or a name of some CommuniGate Pro Domain Alias, otherwise the Server will report the `account has been moved to a remote system` error.

**Server Port**
> The network port the CommuniGate Pro Server uses for MAPI clients. This is the same port as the port used for IMAP clients.

**Use a Secure (SSL/TLS) connection**
> If this option is selected, the MAPI Connector establishes a regular (clear text) network connection to the specified Server port, and uses the STARTTLS command to encrypt all data sent between the workstation and the Server. STARTTLS allows you to use the same server IMAP port for both clear text and secure connections.
> If this option is selected, and the specified Server Port is 993 (the standard port number for secure IMAP), then the MAPI Connector establishes a secure connection to that port. This method should be used if the only port available on the server is the secure IMAP port.
> See the PKI section for more details.

**Use Windows Integrated Authentication (Kerberos)**
> Select this option if your Windows workstation is a member of a Windows/Active Directory Domain or is controlled by some other Kerberos KDC. The MAPI Connector will use your Windows username and credentials to connect to the CommuniGate Pro server.
> If you want to use this option, make sure that:

> - Your CommuniGate Pro Account has the Kerberos Authentication method enabled.
> - Your CommuniGate Pro Domain has the proper Kerberos Keys exported from the Active Directory or Kerberos KDC.

**Use the Following Authentication details**
> Select this option if you want to specify the Account name explicitly.

`Account Name`
> The name of the CommuniGate Pro Account to work with. This name can be a qualified name in the *accountName@domainName* form. If the simple name form is used (the name does not contain the `@` symbol), the MAPI Connector adds the `Server Name` setting value to the specified account name.

`Password`
> The password for the specified CommuniGate Pro Account.

`Remember Password`
> If this option is not selected, the MAPI Connector will present a Login dialog box every time it needs to connect to the Server. If this option is selected, the supplied password is stored in the MAPI Connector settings data.

`Use Secure Authentication`
> If this option is selected, the MAPI Connector sends passwords using secure (encoded) SASL CRAM-MD5 method. The secure method does not work if passwords are stored on the Server using a one-way encrypted method (see the [Security](#) section for more details). In this case this option should be disabled, and the MAPI Connector will send passwords in clear text.
> **Note:** if you need to send passwords in clear text while connecting to the Server via public networks, enable the `Use a Secure connection` option, so all information is encrypted.

## Account Settings

The Account Settings dialog box can be opened by pressing the Account Settings button on the Server panel. The box allows you to specify the MAPI Account name and other general data:



## Connection

The Connection panel allows you to select your network connection method.
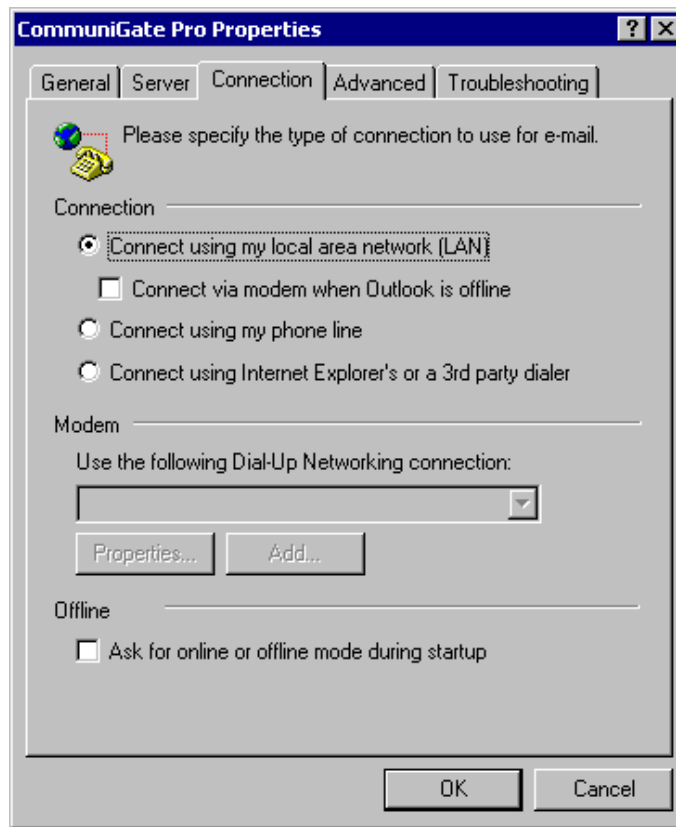
`Offline`
When working in the Offline mode (without a server connection, and using the local message cache only), you may want to tell the Connector to start in the Offline mode.
This will eliminate connection attempts on startup (to avoid bringing up your modem, for example).

## Advanced

The Advanced panel allows you to specify other CommuniGate Pro Accounts you want to work with.
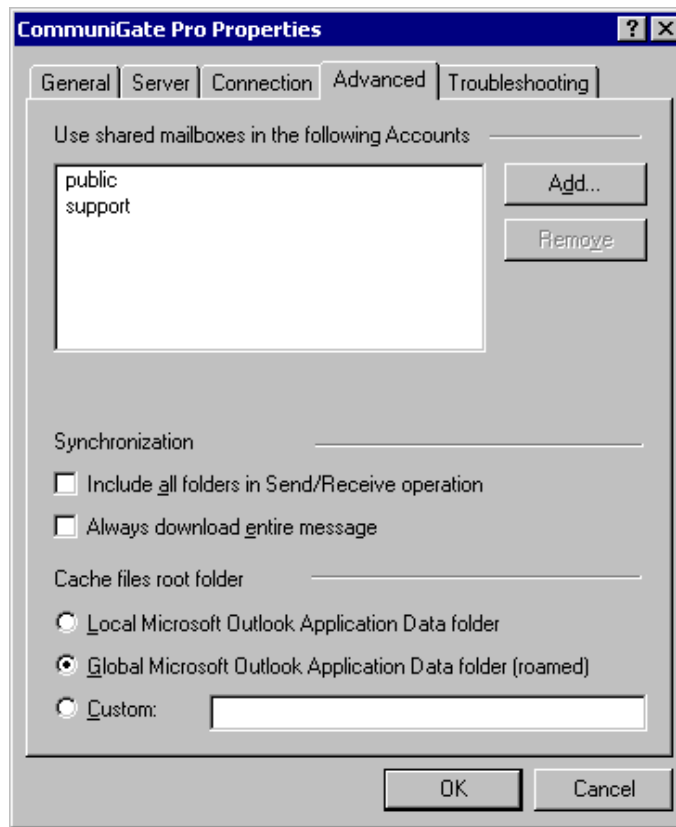
Use the Add and Remove buttons to specify the names of other CommuniGate Pro Accounts. If you want to access an Account in a different Domain, specify the full name: *accountName@domainName*.

The Account owners must grant you Mailbox Access Rights, otherwise you won't be able to see and open Mailboxes in those Accounts. See the [Mailboxes](#) section for more details on foreign Mailbox access.

The Synchronization settings allow you to override the selection of folders to work with in the [Offline mode](#).

```
Include all folders in Send/Receive operation
```
Use this option to select all the account folders for download, regardless the selection in Outlook -> Tools -> Options -> CommuniGate Pro.

```
Always download entire message
```
With this option checked the Connector will download the entire message bodies (versus Headers Only).

Use the Cache files root folder settings to specify where the MAPI Connector should store its local cache. The local cache is used for most MAPI Connector operations, it also allows you to use the MAPI applications (such as Outlook) in the off-line mode.

```
Local Outlook Application Data folder
```
Use this option to store the cache files in the default location on your workstation. The cache file will be available on this workstation only.

```
Global Outlook Application Data folder
```
Use this option is you use Windows roaming features and plan to use your Outlook application from several workstations.

Note: the cache files can be quite large (they may even store a complete copy of all your server-based Mailbox data, depending on the Connector settings). If the cache file becomes large, you can experience delays when you are logging into the workstation, as the workstation needs to copy all roaming data.

```
Custom
```
Use this option if you want to store the cache files in a custom location. For example, you may want to choose a shared folder on a file server to avoid delays caused by roaming.

# Enabling Mailbox Sharing

You can specify Access Control List for your Mailboxes to grant access to those Mailboxes to other CommuniGate Pro users.

Select a Mailbox in the Outlook Folder List, and use the `Properties` menu item to open the Properties dialog box. Open the Permissions panel:
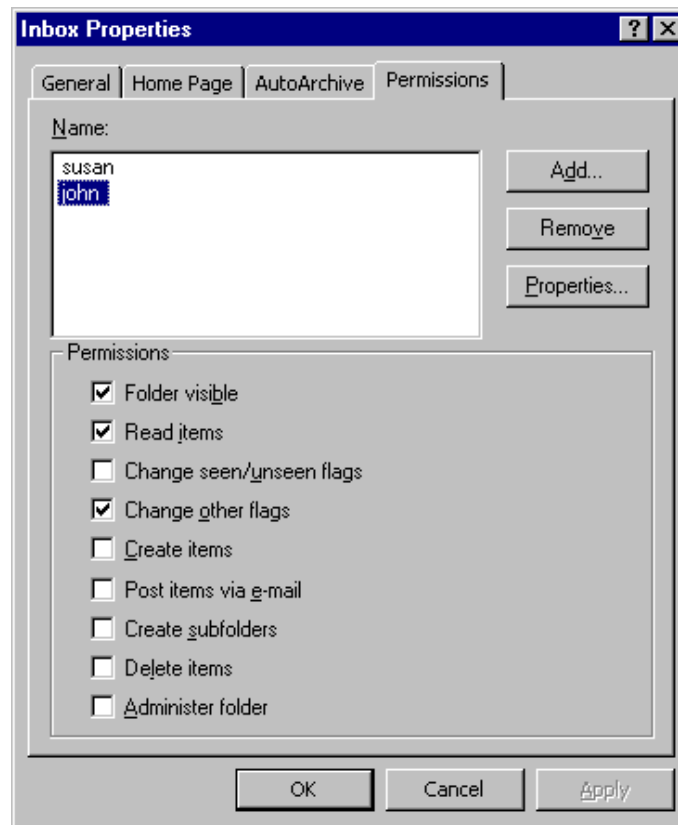


Use the Add and remove buttons to specify the Accounts and other *identifiers* to specify those who should have access to this Mailbox.

Select an identifier in the list and use the checkboxes to grant required access rights to this identifier. See the Mailboxes section for more details on Mailbox ACLs.
**Note:** a user needs to have the Admin Access Right in order to specify the default Mailbox (Folder) View.

# Delegation

The MAPI Connector supports the "delegation" feature: users can send E-mails and Calendar invitations "on behalf of" or "as" other users.

To grant some user delegation rights for your Account, grant that user the Read (Select) right for your INBOX.

# Free/Busy Information

The Free/Busy information is a file specifying when the person is busy, free, out of the office, etc. This information is usually made publicly available, so other users can access it when planning their meetings, scheduling appointments, etc. To compose the Free/Busy data, the groupware client application collects data from user Calendar(s), and merges it into one Free/Busy schedule.

# Posting Free/Busy Information

The MAPI Connector stores your Free/Busy information in your Account [File Storage](#).
Publicly available information in the standard vCalendar format is stored as the `freebusy.vfb` file in the topmost directory of your File Storage.

**Note:** Make sure your CommuniGate Pro [Account Settings](#) allow the MAPI Connector to store a file with your Free/Busy information in your File Storage.

This feature allows users of Outlook and other calendaring clients to access your File Storage via HTTP and retrieve your Free/Busy information. The URL for the CommuniGate Pro Connector user Free/Busy information is

`http://domainName:port/~accountName/freebusy.vfb`

# Accessing Free/Busy Information for Other Users

In order to process Appointments and Meetings, the Outlook application on the client machine should be able to access the Free/Busy information of other users. This operation is implemented via the MAPI Connector, but also may be done by the Outlook application itself. To configure your Outlook application:

- Select Options from the Tools menu to open the Options dialog box.
- Click the Calendar Options button to open the Calendar Options dialog box.
- Click the Free/Busy Options button to open the Free/Busy Options dialog box.
- Enter the `http://%SERVER%/~%NAME%/freebusy.vfb` URL string into the Search field. (read all the notes below).
- Click the OK buttons to close all dialog boxes.

This option will be used by the Outlook application when it needs to retrieve the Free/Busy information for an E-mail user. The application substitutes the %SERVER% symbols with the domain part of the user E-mail, and the %NAME% symbols with the username part of the user E-mail, so for the E-mail address `john@myserver.dom` the Outlook will use the `http://myserver.dom/~john/freebusy.vfb` URL to retrieve John's Free/Busy schedule.

**Note:** the suggested Search URL will work only if your CommuniGate Pro Server accepts WebUser Interface connections on the port 80. If it accepts them on the default port 8100, or on any other non-standard port, the Search URL must include that port:

`http://%SERVER%:8100/~%NAME%/freebusy.vfb`

**Note:** the suggested Search URL will work only if your CommuniGate Pro Domains have names that have A-records pointing to the CommuniGate Pro server. Often, the DNS system does not contain any A-record for your `mydomain.dom` Domains, or those records point to a different system (company Web server), while the CommuniGate Pro Server addresses are specified as `mail.mydomain.com`, or `cgate.mydomain.com`, or `mx.mydomain.com` or similar DNS A-record(s). In this case the Search URL must be modified to use the proper domain names:

`http://mail.%SERVER%/~%NAME%/freebusy.vfb`

**Note:** if your CommuniGate Pro server is serving only one Domain, then you can specify the Search URL as:

`http://mail.mydomain.com/~%NAME%/freebusy.vfb`
where mail.mydomain.com is the name of the CommuniGate Pro Domain or its alias. This should be a name of a DNS A-record pointing to the CommuniGate Pro Server.

Search URLs specified above allow users to retrieve Free/Busy information for the users of the same CommuniGate Pro system.

The Search URL may be used to retrieve the Free/Busy Information for users of other CommuniGate Pro Servers, as long as the Search URL correctly represents their Free/Busy file URLs. To overwrite the Search URL and specify a different (correct) path to some remote user Free/Busy file, create a Contact record for that user, click on the Details tab and enter the correct FreeBusy file URL into the Internet Free-Busy Address field. See the Microsoft Outlook manual for more details on these settings.

# Working Offline

When you work in the Offline mode, the MAPI Connector does not have access to the messages stored on the CommuniGate Pro server. To be able to work productively, you need to make sure that the messages you need are stored in the MAPI Connector local cache. You specify the cache options on the per-Mailbox (per-folder) basis.

Use the Options item in the Outlook Tools menu to open the Options dialog box. Then open the CommuniGate Pro panel:
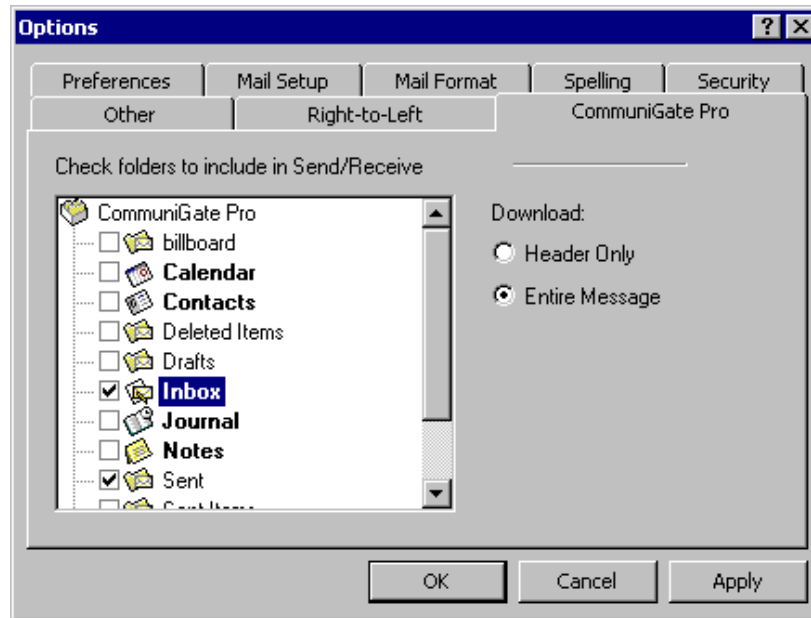


Select the folder you need to work with when you use the Offline mode, and select the downloading method. If you select to download the entire message, the folder name will be shown in bold, if you choose to download message headers only, there will be only a check mark next to the folder name.

Use the Outlook Tools->CommuniGate Pro Server menu to synchronize the changes you do in the Offline mode with the CommuniGate Pro server.
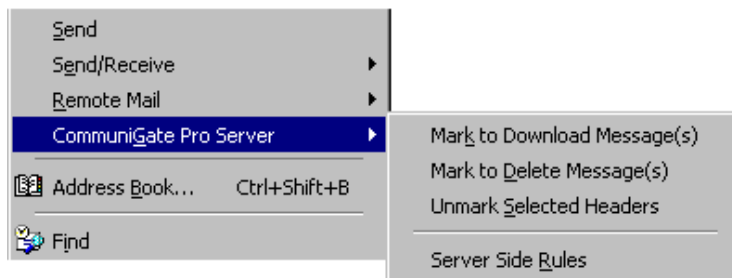


Synchronization takes place when the Send/Receive operation is initiated (manually or automatically, based on the schedule).

`Mark to Download Message(s)`
    The messages selected in the Outlook Mailbox view will be marked so their full bodies are downloaded during the next Send/Receive operation.

`Mark to Delete Message(s)`
    The messages selected in the Outlook Mailbox view will be marked so they will be removed from the Server storage during the next Send/Receive operation.

`Unmark Selected Headers`
    Cancels the pending off-line operations for the messages selected in the Outlook Mailbox view.

# Configuring Automatic Rules

The MAPI Connector allows you to specify the server-side Rules to process mail coming to your Account.

Use the Tools->CommuniGate Pro menu to open the Rules editor window:
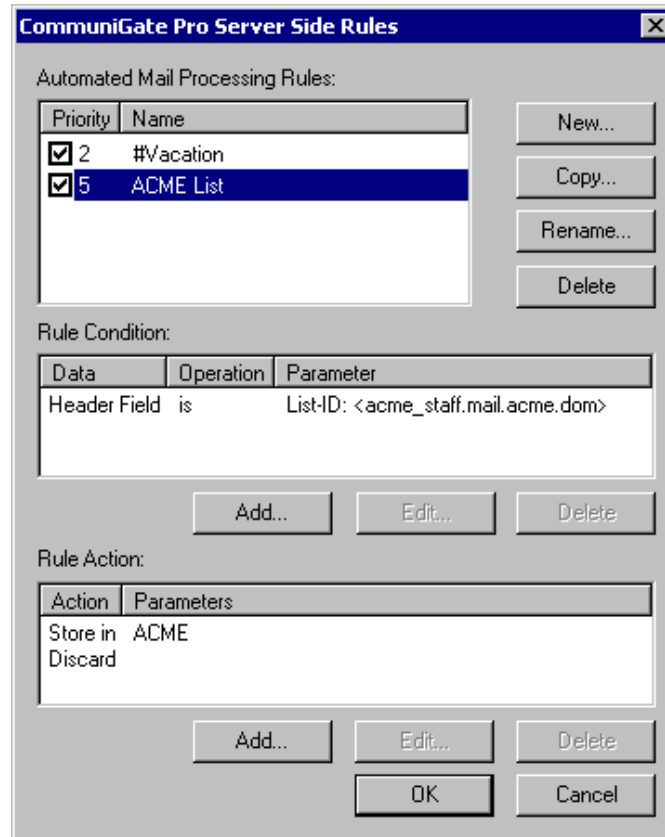


Click the New button to create a new Rule. New Rule has no condition and no action.

Click the Add button to add Rule conditions and actions:
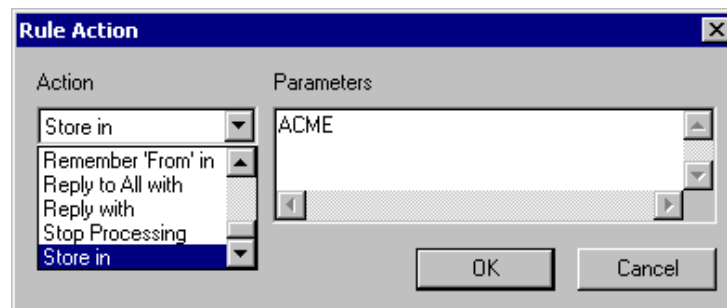


See the Automatic Rules section for more details.

---

# WebMail Integration

The MAPI Connector employs the user WebMail (WebUser Interface) settings. It instructs the MAPI applications (such as the Microsoft Outlook) to use the same names for "Special" Mailboxes. As a result MAPI applications and the WebUser Interface use the same Trash or Deleted Items Mailbox to store removed messages, use the same Mailbox as the Main Calendar Mailbox, etc.

The MAPI Connector also retrieves the user Domain Mail Trailer setting. The content of that setting is added to all non-

encrypted and not-signed text messages submitted via the MAPI Connector.

The values specified via the Account Settings panel are stored in the WebUser Settings, so both the WebUser Interface and MAPI sessions use the same From:, Reply-To:, and Organization values.

# Communicating with Microsoft Exchange users

Outlook users that work with Exchange servers may have problems sending meeting requests to your users working with the CommuniGate Pro MAPI Connector. Meeting requests sent via an Exchange server may come in as plain text messages instead. The Exchange users should adjust the configuration of their Outlook applications:

> Using Outlook Tools menu, Exchange users should open the Options dialog box. After they click the Calendaring Options button, the dialog box appears and they should enable the `Send meeting requests using iCalendar by default` option.

# Real-Time Communications

The MAPI Connector allows Outlook users to employ Real-Time (VoIP, etc.) functions of the CommuniGate Pro Server.

Outlook users can initiate a phone call using the telephone number specified in a Contact item. Right-click a Contact record to open a pop-up menu and select the Call using CommuniGate Pro item:
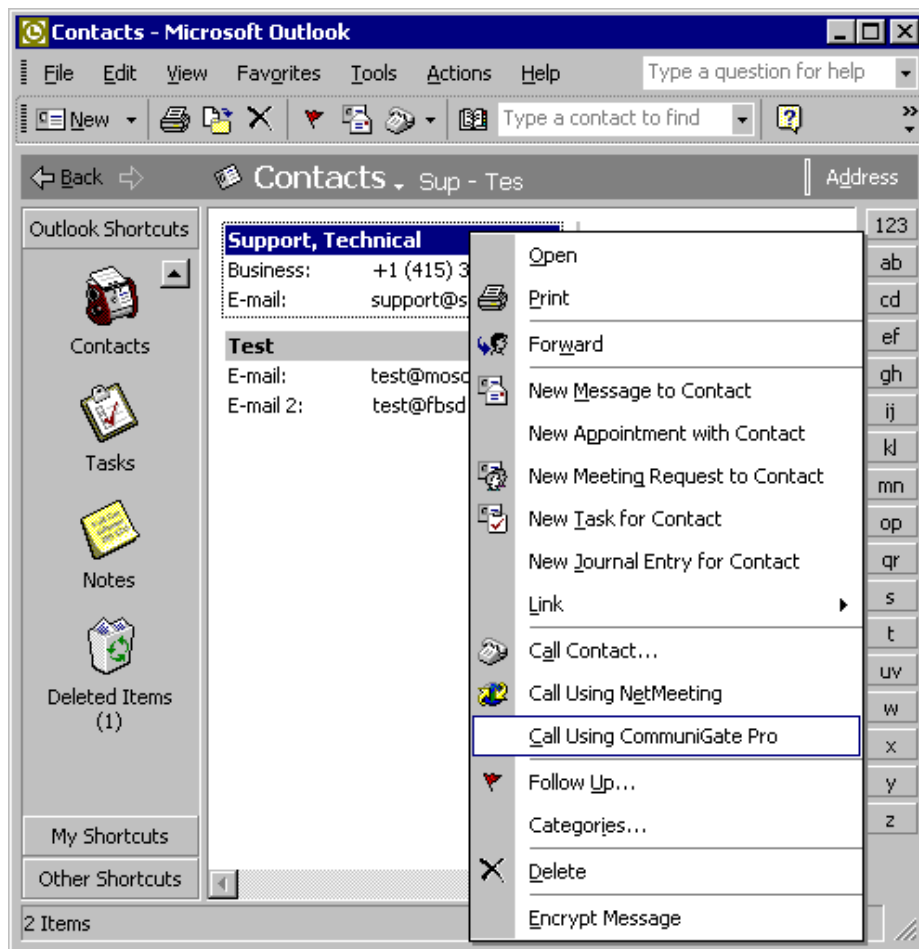


You can also use the CommuniGate Pro Server submenu in the Tools menu.

A dialog box with the Contact name and phone number appears:
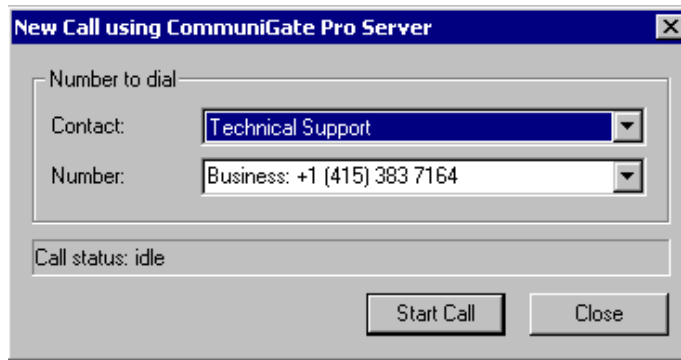
You can type a different phone number in the second field. That number will be used for this call only, it will not be stored with the Contact Item.

Click the Start Call button to initiate a call. All your SIP devices will start to ring immediately. Answer this call on any device, and the Server will instruct it to initiate a call to the selected phone number.

The dialog box has a field displaying the call status:
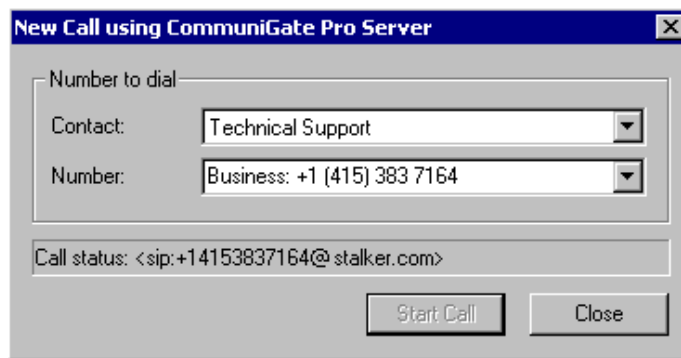


# Server-side Encryption

The MAPI connector allows you to specify Server-side Rules, including the Rules that can store incoming messages in an encrypted form.

You may also want to improve security for certain messages received and stored on the Server in clear-text.

Right-click on the selected message in Outlook. A pop-up menu will open:
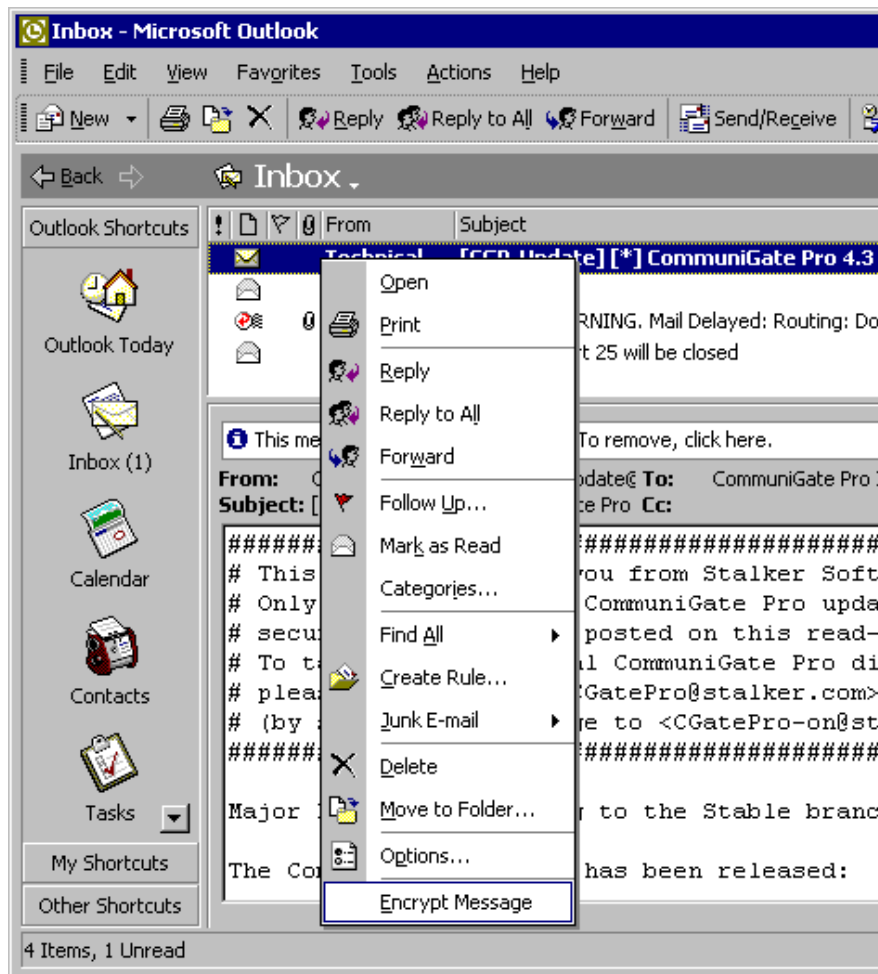
When you select the Encrypt Message item, the MAPI Connector will send your default Certificate (containing your Public Key) to the Server. The Server will use that Certificate to encrypt the selected message. It will be stored on the Server in the encrypted S/MIME format - as if the message sender has sent it encrypted.

If you want to read Encrypted messages using your MAPI clients (Outlook) and the WebUser Interface, make sure that your Windows desktop system and the WebUser Interface have the same Private Keys and Certificates installed. You can either generate a key and Certificate in the WebUser Interface and then export them to your Windows desktop system, or you can export your keys from the Windows desktop system and import them into your WebUser Interface settings.

# Troubleshooting

The MAPI connector works as a liaison between MAPI applications (such as Microsoft Outlook) and the CommuniGate Pro Server. The problem a user can experience with its client, can be a bug or feature of that client, or a problem in the MAPI Connector or Server software. To help investigate the problem, the MAPI Connector can generate a detailed Log of all operations it was requested to perform. You can examine that Log yourself or send it to CommuniGate Systems technical support.

Open the Troubleshooting panel in the MAPI Connector ("CommuniGate Pro Service") setup box:
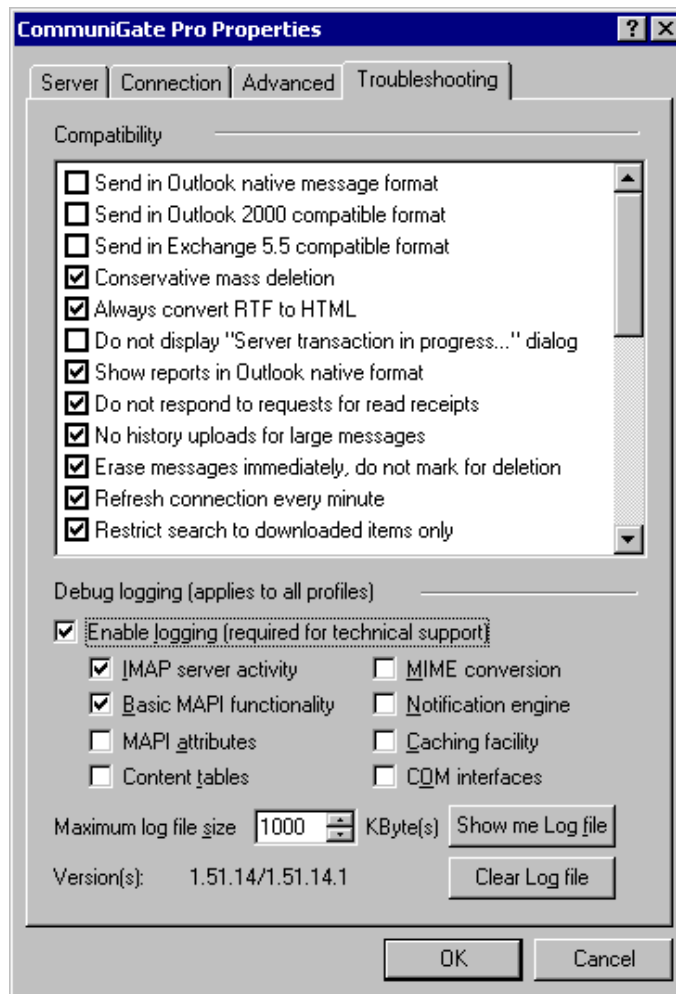
The panel displays the versions of both MAPI Connector software components: the *starter code* library installed on your desktop computer and the *server code* library retrieved from the CommuniGate Pro Server.

Select the Enable Logging option to start MAPI Connector log recording. The MAPI Connector Log keeps only the last records, so the Log file does not exceed the size specified in the Maximum Log File Size setting.

Use the checkbox controls to enable logging for various MAPI Connector components.

Click the Show Me Log File button to open a file directory window the Log file is stored in. You can use this feature to E-mail the Log file to CommuniGate Systems technical support.

Click the Delete Log File to clear the Log file.

Use the Compatibility options to tune up the MAPI Connector operation for mixed environments.

```
Send in Outlook native message format
```
> In some rare cases special messages (such as Task requests or forwarded Contacts) sent via the MAPI Connector in the Internet format may be interpreted incorrectly by Outlook Exchange (or Outlook IMAP) users. Checking this option may help to solve the problem.

```
Send in Outlook 2000 compatible format
```
> Older versions of Outlook may have problems interpreting Calendar and Contact items in the format used by newer versions of Outlook. Check this option to use older format for these items.

```
Send in Exchange 5.5 compatible format
```
> Meeting requests in formats used by newer versions of Outlook may be incorrectly interpreted by Exchange 5.5 servers. Check this option if any of your correspondents use this version of Exchange server and have problems accepting your meeting requests.

```
Conservative mass deletion
```
> In some configuration the Outlook Autoarchive function may remove messages in large portions. To avoid this problem, select this option.

```
Always convert RTF to HTML
```

Selecting this option causes the `text/rtf` MIME parts in outgoing messages to be converted to HTML, which is understood by the larger number of mail client applications.

`Do not display "Server transaction in progress..." dialog`

During long server transactions the Connector displays a progress indicator window. If you don't want that select this option.

`Show reports in Outlook native format`

Outlook can use a special form to display *non delivery notification* messages, which provides interface for re-sending of the failed messages. For this feature to work correctly the non-delivery report should contain the full body of the failed message, which is not always the case.

`Do not respond to requests for read receipts`

The sender of a message may request a receipt to be sent back to him when you open a message for reading. To ignore such requests select this option.

`No history uploads for large messages`

To fulfill Outlook requests to set some extended property on a message the MAPI connector may re-upload modified message back to the server. This may take quite a while for large messages. To disable this kind of activity check this option.

`Erase messages immediately, do not mark for deletion`

To increase performance, by default messages are not removed immediately but rather are marked for deletion and removed physically when the folder containing them is closed. If the folder is accessed by several clients simultaneously, these items marked for deletion may still appear in those clients and confuse users. To avoid that you can check this option.

`Refresh connection every minute`

Some NAT and anti-virus/spam firewalls are very aggressive in terminating TCP connections which they assume to be idle. Closing IMAP sessions this way on the TCP level may confuse Outlook. With this option checked, the MAPI Connector will produce some minimal data exchange on the connection to the CommuniGate pro server, so that connection does not appear idle to the firewall.

`Restrict search to downloaded items only`

Searching through some message attributes may require downloading items to the client. This may take quite a while for large message store. With this option checked only those items that were previously cached by the MAPI Connector will be searched.

# Known Limitations

The protocols and APIs the MAPI Connector implements are not the Internet standards, but API from the Microsoft® Corporation. Those APIs are not fully documented, and as a result, some minor client (Outlook) functions may be unavailable. CommuniGate Systems is working on solving these problems, and the MAPI Connector updates are released on a regular basis.

The special Known MAPI Problems site lists all reported and known problems and provides the status update on them. Please consult this site first if you have any problem with the MAPI Connector functionality.

Overview  POP  IMAP  WebUser  XIMSS  MAPI  AirSync  WebDAV  FTP  TFTP  ACAP

# AirSync Module

- **Configuring the AirSync Module**
- **Restricting Access**
- **Managing AirSync Devices**
- **Compatibility Notes**

The CommuniGate Pro Server can be used with Microsoft® Windows Mobile and other compatible devices (PDAs, smartphones) supporting the AirSync protocol - a client-server version of the Microsoft ActiveSync protocol.

The AirSync protocol allows a client device to synchronize its Calendaring, Tasks, and Contacts data with the same data in the user Account on the server. The protocol also allows a user to synchronize some or all of its E-mail mailboxes, to compose and send E-mails, Event invitations, and Event replies.

The CommuniGate Pro Mailbox Alias feature allows AirSync clients to access shared Mailboxes in other Accounts.

The AirSync protocol allows a client device to access the Server Directory.

# Configuring the AirSync Module

The AirSync protocol works over the HTTP protocol, using the HTTP User Module. Open the HTTP User Module settings, and find the Sub-Protocols panel:

| Sub-Protocols | |
|---|---|
| Access | Log Level |
| AIRSYNC: clients | Low Level |

Use the AIRSYNC Log Level setting to specify the type of information the AirSync module should put in the Server Log. The AirSync module records in the System Log are marked with the AIRSYNC tag.

The Access setting specifies who can access the AirSync service.

**Note:** Many AirSync clients are designed to access only the standard HTTP ports (port 80 for clear-text connections and port 443 for secure connections), make sure your HTTP User module is properly configured to accept connections on these ports.

# Restricting Access

Server and Domain administrators may want to restrict AirSync access to to certain devices only:

| Enabled Mobile Device IDs |
|---|
| !phoneX*, !vendor1/*, * |

The setting value can contain zero, one, or several elements separated using the comma (`,`) symbol:

- if no element is specified, access is prohibited to all devices
- each elements is compared to the ID of the device trying to connect; if an elements contains the `*` symbols, they are used as wildcards; The element `*` grants access to all devices
- if an element contains the `/` symbol, the element part before that symbol is compared to the device type (such as `PocketPC,Apple`), and the part after the slash symbol is compared to the device ID
- if an element starts with the `!` symbol and the rest of the element matches the device ID (or the device type and the device ID), the device is prohibited from accessing the Account(s)

In the example above, the Default Setting contains the following elements:

`!phoneX*`
> devices having IDs starting with `phoneX` are prohibited

`!vendor1/*`
> devices having the `vendor1` device type are prohibited

`*`
> all other devices are allowed

In the example above, the custom Setting (overriding the Default Settings) contains the following elements:

`9999BFE4D66C4BE29AAB1DF8F8CF4064`
> the device with this ID is allowed

`android678263333`
> the device with this ID is allowed

All other devices are prohibited.

# Managing AirSync Devices

Server and Domain administrators can monitor and manage AirSync devices employed by the Account user.

Open the Account Settings WebAdmin Interface page, open the Mail section, and follow the Mobile link. Find the AirSync Devices panel:

| AirSync Clients | | | | |
|---|---|---|---|---|
| **Client Type** | **ID** | **Policy** | **Last Access** | **Remote Device Wipe** |
| PocketPC | 9999BFE4D66C4BE29AAB1DF8F8CF4064 | {UsePIN=YES;} | 10:54:48PM | 1-Oct-08 |
| PocketPC | FAFABFE4D3333BE2ABBC2E0909D05175 | {} | 20-Sep | pending |

The table elements show the device type and unique ID for each device the Account user has used, as well as the time of the last AirSync operation with that device.

Devices can be protected with a PIN (password) that needs to be entered every time the device is switched on, or after a period of inactivity. You can remotely enable or disable this function: select the device element and click the Enable PIN-lock or Disable PIN-lock button.

To remove all information from the device (if it has been lost or stolen), select the device element and click the Wipe Out button.
The device element will display the `pending` marker.
Next time the device tries to connect to the Server, it will receive a command instructing it to remove all E-mail and other data stored locally on the device.

To cancel the Wipe Out operation (if the device has been recovered before it tried to access the Server), select the device element and click the Cancel button.

Some devices may fail to deal with all user Mailboxes - either because the user Account has too many Mailboxes, or because some Mailbox has too many items in it. A user can "hide" certain Mailboxes from the AirSync client by disabling the Visible to Mobile Devices Mailbox option.

# Compatibility Notes

The AirSync protocol was designed to work in a proprietary, non-standard environment. CommuniGate Pro AirSync module makes its best effort to convert AirSync data to the standard formats (such as iCalendar and vCard).

## Replying to Event Requests

When an AirSync client accepts or declines an Event, it generates an E-mail message to be sent to the Event organizer. This message contains Microsoft proprietary data not suitable for any standard-based application.
To provide a workaround for this problem, the CommuniGate Pro AirSync module generates a Reply itself, using the standard iCalendar format. As a result, the organizer may receive 2 Event Reply messages - one from the AirSync module, and one from the Windows Mobile client itself.
**Note:** when the Windows Mobile user chooses to accept or decline a request without sending a reply, only the Windows Mobile own request is affected. The AirSync module generates its reply independently, based only on the Event request "RSVP" attribute.

## Sub-Mailboxes

AirSync clients do not support sub-mailboxes of the *Sent* and *Trash* Mailboxes, and older clients do not support sub-mailboxes of the *INBOX* Mailbox either.
If an Account contains such sub-mailboxes, the Server does not inform an AirSync client about them.

Overview POP IMAP WebUser XIMSS MAPI AirSync **WebDAV** FTP TFTP ACAP

# WebDAV Module

- **Root Element**
  - **Configuring the Root Element DAV Module**
- **Principal Elements**
  - **Configuring the Prinicipal Elements DAV Module**
- **File Access**
  - **Configuring the FileDAV Module**
- **CalDAV**
  - **Configuring the CalDAV Module**
  - **Using the Publish/Subscribe (ICS) method.**
- **CardDAV**
  - **Configuring the CardDAV Module**

The WebDAV protocol is an extension of the HTTP protocol. Various clients use it to access and manage various data resources.
The CommuniGate Pro WebDAV module implements authenticated access to Account File Storage, Calendar, Task, and Contacts mailboxes.

The CommuniGate Pro WebDAV implementation is a part of the HTTP User module, and that module settings are applied to the WebDAV functions, too.

# Root Element

The WebDAV component provides a read-only "root" element, with the following URLs:
`http://servername[:port]/`
or
`https://servername[:port]/` URL
where `port` is a HTTP User Module port (8100, 9100 by default).

Various WebDAV (CalDAV, CardDAV, etc.) clients can be configured to use this Root Element. They can retrieve the Root Element properties, which direct them to other URLs (realms), implementing the respective protocols.

## Configuring the Root Element Module

The WebDAV protocol works over the HTTP protocol, using the HTTP User Module. Open the HTTP User Module settings, and find the Sub-Protocols panel:

| Sub-Protocols | |
|---|---|
| **Access** | **Log Level** |
| RootDAV:    clients | Low Level |

Use the RootDAV Log Level setting to specify the type of information the Root Element DAV module should put in the Server Log.
The Root Element DAV module records in the System Log are marked with the `RootDAV` tag.

The Access setting specifies who can access the RootDAV service.

# Principal Elements

The WebDAV protocol and protocols based on WebDAV allow users to access resources of other users if the proper access rights have been granted.
To support WebDAV ACLs (Access Control Lists), each CommuniGate Pro users is granted a unique URL that represents that user in the Access Control Lists. These URLs are:
`http://servername[:port]/UserDAV/accountName@domainName`
or
`https://servername[:port]/UserDAV/accountName@domainName/` URL
where `port` is a [HTTP User](#) Module port (8100, 9100 by default).

## Configuring the Principal Elements DAV Module

The WebDAV protocol works over the [HTTP](#) protocol, using the HTTP User Module. Open the HTTP User Module settings, and find the Sub-Protocols panel:

| Sub-Protocols | |
|---|---|
| **Access** | **Log Level** |
| UserDAV:   clients | Low Level |

Use the UserDAV Log Level setting to specify the type of information the Principal Elements DAV module should put in the Server Log.
The Principal Elements DAV module records in the System Log are marked with the `UserDAV` tag.

The Access setting specifies who can access the Principal Elements DAV Module service.

# File Access

The CommuniGate Pro Server implements access to [Account File Storage](#) via the WebDAV protocol (FileDAV).

All FileDAV data requests must be authenticated: the user should specify the Account name and password. The Account and its Domain must have the WebSite Service enabled.

The "root" of the user File Storage has the following URL:
`http://servername[:port]/WebDAV/`
or
`https://servername[:port]/WebDAV/`
where `port` is a [HTTP User](#) Module port (8100, 9100 by default).

To Access the File Storage of some other user, the following URLs should be used:


`http://servername[:port]/WebDAV/~accountName/`
or
`https://servername[:port]/WebDAV/~accountName/`
where `port` is a [HTTP User](#) Module port (8100, 9100 by default), and the *accountName* is the name of the Account to access. If this Account is located in a different Domain, use *accountName@domainName* instead.

## Configuring the FileDAV Module

The WebDAV protocol works over the [HTTP](#) protocol, using the HTTP User Module. Open the HTTP User Module settings, and find the Sub-Protocols panel:

| Sub-Protocols | |
| --- | --- |
| **Access** | **Log Level** |
| FileDAV: clients | Low Level |

Use the FileDAV Log Level setting to specify the type of information the FileDAV module should put in the Server Log. The FileDAV module records in the System Log are marked with the `FileDAV` tag.

The Access setting specifies who can access the FileDAV service.

---

# CalDAV

The CommuniGate Pro Server supports the CalDAV protocol, a WebDAV protocol extension.

This protocol allows CalDAV clients to access and modify Calendaring and Tasks data stored in the user Account(s).

A full-featured CalDAV client can detect the proper URL for CalDAV operations itself, when it is directed to the WebDAV Root element:
`http://`*servername*`[:`*port*`]/`
or
`https://`*servername*`[:`*port*`]/` URL
where `port` is a [HTTP User](#) Module port (8100, 9100 by default).

If the CalDAV client cannot detect the CalDAV URL location, it should be specified explicitly:

> `http://`*servername*`:`*port*`/CalDAV/`

> `https://`*servername*`:`*port*`/CalDAV/`

Some CalDAV clients need to be configured to access a particular Calendar or Task collection. Any Calendaring or Tasks Mailbox can be accessed using the following URLs:

> `http://`*servername*`:`*port*`/CalDAV/`*mailboxName*`/`

> `https://`*servername*`:`*port*`/CalDAV/`*mailboxName*`/`

If the mailboxName ends with the `ics` file extension, the Server removes that extension.

All CalDAV data requests must be authenticated: the user should specify the Account name and password. The Account and its Domain must have the WebCal Service enabled.

If the user Domain Name or Domain Alias name is mail.company.com, the HTTP User port is 80, and the Mailbox name is `Calendar`, the access URL is

> `http://mail.company.com/CalDAV/Calendar`

or

> `https://mail.company.com/CalDAV/Calendar.ics`

Any Calendar-type or Tasks-type Mailbox can be accessed this way. To access a Mailbox in a different Account, the full Mailbox name should be specified:

> `http://mail.company.com/CalDAV/~username/Calendar`

The authenticated user should have proper [access rights](#) to retrieve and/or modify data in Mailboxes belonging to other users.

The CalDAV module checks the [Mailbox Subscription](#) list and reports all Calendar-type Mailboxes in other Accounts included into this list.

The CalDAV module stores Calendar and Tasks attachments in the Account [File Storage](#), inside the `private/caldav/` directory.

# Configuring the CalDAV Module

The CalDAV protocol works over the [HTTP](#) protocol, using the HTTP User Module. Open the HTTP User Module settings, and find the Sub-Protocols panel:

| Sub-Protocols | |
|---|---|
| **Access** | **Log Level** |
| CalDAV:  clients | Low Level |

Use the CalDAV Log Level setting to specify the type of information the CalDAV module should put in the Server Log. The CalDAV module records in the System Log are marked with the `CalDAV` tag.

The Access setting specifies who can access the CalDAV service.

# Using the Publish/Subscribe (ICS) method

The CommuniGate Pro HTTP User module implements an older, pre-CalDAV method of dealing with data in Calendar and Task Mailboxes.
The module allows client applications to retrieve all items using the iCalendar format. This operation is often called *subscribing* to Calendar data.
The module also allows client applications to rewrite the Mailbox content new iCalendar-formatted data (to *publish* calendaring data).

The URL for Calendaring data is:

```
http://servername:port/CalendarData/mailboxName
```

```
https://servername:port/CalendarData/mailboxName
```

where `port` is the HTTP User port (8100, 9100 by default).
If the mailboxName ends with the `ics` file extension, the server removes that extension.
If the mailboxName is empty, the Server uses the Default Calendar name.

All Calendaring data requests must be authenticated: the user should specify the Account name and password. The Account and its Domain must have their WebCal Service enabled.

If the user Domain Name or Domain Alias name is mail.company.com, the HTTP User port is 80, and the Mailbox name is `Calendar`, the access URL is

```
http://mail.company.com/CalendarData/Calendar
```

or

```
http://mail.company.com/CalendarData/Calendar.ics
```

Any Calendar-type or Tasks-type Mailbox can be accessed this way. To access a Mailbox in a different Account, the full Mailbox name should be specified:

```
http://mail.company.com/CalendarData/~username/Calendar.ics
```

The authenticated user should have proper [access rights](#) to retrieve and/or modify data in Mailboxes belonging to other users.

The HTTP module supports the following HTTP operations for the /CalendarData/ realm:

- GET/HEAD: the Mailbox is parsed, and all items containing iCalendar information are retrieved as one VCALENDAR object with VEVENT and VTODO elements. If the Mailbox is a foreign one, the authenticated user must have the `Select` access right for that Mailbox.
- DELETE: the Mailbox is parsed and all items containing iCalendar information are removed. If the Mailbox is a foreign one, the authenticated user must have the Delete access right for that Mailbox.
- PUT: the HTTP request body is parsed as a VCALENDAR object. All parsed VEVENT and VTODO elements are added

to the Mailbox as separate items. If parsing of any element failed, no item is added. If the Mailbox is a foreign one, the authenticated user must have the Insert access right for that Mailbox.

Some applications do not support the DELETE method. These applications expect that the PUT operation removes all previous information from the Calendar Mailbox.
To support these applications, use the `CalendarDataDel` realm instead of the `CalendarData` realm, or include the `DeleteAll="1"` parameter into the URL.
In this case each PUT operation will be preceded with a virtual DELETE operation removing all existing iCalendar items from the Mailbox.

# CardDAV

The CommuniGate Pro Server supports the CardDAV protocol, a WebDAV protocol extension.

This protocol allows CardDAV clients to access and modify Contacts data stored in the user Account(s).

A full-featured CardDAV client can detect the proper URL for CardDAV operations itself, when it is directed to the WebDAV Root element:
`http://`*servername*`[:`*port*`]/`
or
`https://`*servername*`[:`*port*`]/` URL
where `port` is a [HTTP User](#) Module port (8100, 9100 by default).

If the CardDAV client cannot detect the CardDAV URL location, it should be specified explicitly:

`http://`*servername*`:`*port*`/CardDAV/`

`https://`*servername*`:`*port*`/CardDAV/`

Some CardDAV clients need to be configured to access a particular Contacts collection. Any Contacts Mailbox can be accessed using the following URLs:

`http://`*servername*`:`*port*`/CardDAV/`*mailboxName*`/`

`https://`*servername*`:`*port*`/CardDAV/`*mailboxName*`/`

If the mailboxName ends with the `vcf` file extension, the Server removes that extension.

All CardDAV data requests must be authenticated: the user should specify the Account name and password. The Account and its Domain must have the WebSite Service enabled.

If the user Domain Name or Domain Alias name is mail.company.com, the HTTP User port is 80, and the Mailbox name is `Contacts`, the access URL is

`http://mail.company.com/CardDAV/Contacts`

or

`https://mail.company.com/CardDAV/Contacts.vcf`

Any Contacts-type Mailbox can be accessed this way. To access a Mailbox in a different Account, the full Mailbox name should be specified:

`http://mail.company.com/CardDAV/~username/Contacts`

The authenticated user should have proper [access rights](#) to retrieve and/or modify data in Mailboxes belonging to other users.

The CardDAV module checks the [Mailbox Subscription](#) list and reports all Contacts-type Mailboxes in other Accounts included into this list.

# Configuring the CardDAV Module

The CardDAV protocol works over the [HTTP](#) protocol, using the HTTP User Module. Open the HTTP User Module settings, and find the Sub-Protocols panel:

| Sub-Protocols | |
|---|---|
| **Access** | **Log Level** |
| CardDAV:   clients | Low Level |

Use the CardDAV Log Level setting to specify the type of information the CardDAV module should put in the Server Log. The CardDAV module records in the System Log are marked with the `CardDAV` tag.

The Access setting specifies who can access the CalDAV service.

---

Overview  POP  IMAP  WebUser  XIMSS  MAPI  AirSync  WebDAV  FTP  TFTP  ACAP

# FTP Module

- **File Transfer Protocol**
- **Configuring the FTP module**
- **Access to Account File Storage**
- **Passive Mode Connections**

The CommuniGate Pro FTP module implements an FTP server for TCP/IP networks.

The FTP protocol allows an FTP client application to connect to the Server computer and specify the user (Account) name and the password. If access to the specified user Account is granted, the client application can retrieve and update data inside that Account File Storage.

# File Transfer Protocol

The File Transfer Protocol allows client computers to work with files stored on remote servers. A computer running an FTP client application connects to the server computer and provides account (user) name and the password. If access to the specified user account is granted, the client application sends protocol commands to the FTP server. These protocol commands tell the server to list all files in the current directory, to change the current directory, to retrieve, upload, rename, and remove files stored on the FTP server.

The CommuniGate Pro FTP module supports all related Internet standards (RFCs).

The CommuniGate Pro FTP module supports the REST command and it can resume broken file transfer operations.

The CommuniGate Pro FTP module supports the GSSAPI authentication method. It can use the established GSSAPI "context" for encryption and integrity protection of the control and data channels.

The CommuniGate Pro FTP module supports the STLS command, as well as non-standard AUTH SSL and AUTH TLS-P commands for establishing secure (TLS) communication links.

# Configuring the FTP module

Use the WebAdmin Interface to configure the FTP module. Open the Access page in the Settings realm.

| Processing |
| --- |

| Log Level: | Major & Failures | Channels: | 10 | **Listener** |
|---|---|---|---|---|
| Passive Mode: | Disabled | | Send WAN Address | Legacy-Style LIST |
| | | | Use Fixed Active Port | |

**Log**

Use this setting to specify what kind of information the FTP module should put in the Server Log. Usually you should use the `Major` (password modification reports) or `Problems` (non-fatal errors) levels. But when you experience problems with the FTP module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well. Most FTP clients send passwords in the clear text format, and setting the Log setting to these values for long periods of time can become a security hole if the Log file can be copied from the Server computer.

The FTP module records in the System Log are marked with the `FTP` tag.

**Channels**

When you specify a non-zero value for the `TCP/IP` Channels setting, the FTP module creates a so-called "listener" on the specified port(s). The module starts to accept FTP connections from FTP clients. This setting is used to limit the number of simultaneous connections the FTP module can accept. If there are too many incoming connections open, the module will reject new connections, and the users should retry later.

If the number of channels is set to zero, the FTP module closes the listener and releases (unbinds from) the TCP port(s).

**Listener**

By default, the FTP module Listener accepts clear text connections on the TCP port 8021. Follow the listener link to tune the FTP Listener.
If the server computer does not have any other FTP server software running, you may want to switch the FTP Listener to the port 21 (the standard FTP port).
**Note:** The FTP protocol has a "NAT traversal" problem. When working in the "active" mode, the FTP server needs to open data connections to the client computer, and if there is a NAT device between the FTP server and the client computer, attempts to establish these data connections would fail. To solve this problem, most NAT devices/programs implement an FTP proxy, but they activate this feature only if they detect an outgoing connection to the port 21.
If you use the FTP module with a non-standard port number (such as 8021), your users connecting from behind NAT devices won't be able to do data transfers in the "active" mode (the "passive" mode should work correctly).

**Passive Mode**

When this option is disabled, the FTP module rejects requests for passive-mode file transfers.

**Send WAN Address**

Use this option to send the Server or Cluster WAN Address when a client requests a Passive Mode transfer.

**Use Fixed Active Port**

If this option is enabled, the FTP module uses the fix TCP port number for active (outgoing) data connections. That port number is the port number the FTP control connection is accepted on minus 1.
If this option is disabled, the FTP module selects the TCP port for active (outgoing) data connections from the TCP port range set using the Network settings. You may want to disable this option for certain Cluster configurations.

**Legacy-Style LIST**

When this option is enabled, the Server always sends a positive response to the LIST command, even if the target directory is unavailable.
This option can help some clients that always open a data connection for LIST results, ignoring error messages the Server sends.

---

# Access to Account File Storage

When an FTP user is authenticated, the current directory is set to the topmost directory of the Account File Storage.
The FTP module allows a user to upload, download, rename and remove file from File Storage and its directories.

The FTP module allows a user to create, remove, and rename directories in the Account File Storage.

It is possible to access File Storage of some other Account by using the `~accountName/` name prefix (to access the *accountName* Account in the same Domain), or by using the `~accountName@domainName/` name prefix to access File Storage of any Account in any Domain.

Please see the [File Storage](#) section for the details on the required Access Rights.

# Passive Mode Connections

The FTP module supports Passive Mode transfers. In this mode, the FTP module opens a separate listener port/socket, sends the IP address and port number of that socket to the client, and the client opens a TCP connection to the specified address and port.

When the CommuniGate Pro Server is located behind a NAT/Firewall, external (WAN) clients using the Passive Mode connect to an external WAN address, rather than the Server own IP address. If the NAT/Firewall cannot fix this problem, use the Send WAN Address option.

The FTP Module uses the [TCP Media Proxy](#) ports for Passive Mode transfers.

CommuniGate Pro

Overview POP IMAP WebUser XIMSS MAPI AirSync WebDAV FTP **TFTP** ACAP

# TFTP Module

- **Trivial File Transfer Protocol**
- **Configuring the TFTP module**
- **Access to Account File Storage**

The CommuniGate Pro TFTP module implements a TFTP server for UDP/IP networks.

The TFTP protocol allows a TFTP client application to retrieve files from the Server computer and to store files on the Server computer. The CommuniGate Pro TFTP clients can read and write Account File Storage files.

# Trivial File Transfer Protocol

The Trivial File Transfer Protocol allows client computers to work with files stored on remote servers. A computer running a TFTP client application sends UDP request packets to the server computer. These packets contain the name of the file to read or to store, and the transfer mode.
For a file read operation, the server computer replies with a UDP packet with a block of file data. If the file is larger than one block, then the client computer sends an ACK (acknowledgment) packet, and the server computer sends the next block of file data in response.
For a file write operation, the server computer replies with an ACK UDP packet, the client computer sends the first file data block, the server computer replies with an ACK packet, and the client computer sends the next data block.

The CommuniGate Pro TFTP module supports relevant Internet standards (RFCs).

# Configuring the TFTP module

Use the WebAdmin Interface to configure the TFTP module. Open the Access pages in the Settings realm, and open the TFTP page:

| Processing | | |
|---|---|---|
| Log Level: Problems | | **Listener** |
| Default Storage: | | Try IP-Address Directory |

Log

Use this setting to specify what kind of information the TFTP module should put in the Server Log. Usually you should use the `Major` (password modification reports) or `Problems` (non-fatal errors) levels. But when you experience problems with the TFTP module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well.

The TFTP module records in the System Log are marked with the `TFTP` tag.

listener

Use this link to open the UDP Listener page and specify the port number and local network address for the TFTP service, and access restrictions for that port. When the port number is set to 0, the TFTP server is disabled.

By default TFTP clients send requests to the UDP port 69.

If your server computer is already running some TFTP server, you may want to specify a non-standard port number here and reconfigure your TFTP client software to use that port number.

Default Storage

Since the TFTP protocol does not authenticate clients, you need to specify the File Storage to be used by default.

Specify a name of an existing Account in this field.

If that Account does not belong to the Main Domain, specify the full Account name as *accountName@domainName*.

You can specify a subdirectory of the Account File Storage by adding the subdirectory name separated with the slash (`/`) symbol: *accountName/directoryName* or *accountName@domainName/directoryName*

Try IP-Address Directory

If this option is enabled, the module adds the client IP address to the specified file name, thus allowing different identically configured clients to work with different file sets (see below).

Run Sessions on Controller

This option is available in a [Dynamic Cluster](#) only.

When this option is enabled, the Server sends all TFTP requests to the Cluster Controller (unless this Server is the active Controller itself), using the inter-cluster CLI protocol. It then relays the Controller responses to the client.

This feature is required when you use a Load Balancer that does not keep any "session" or "state" for UDP requests, and subsequent requests within the same TFTP session can be directed to different Cluster members.

# Access to Account File Storage

The file name specified in the TFTP read or write request packet is interpreted as the name of a file in the Default Account File Storage.

If the specified file name starts with the slash (`/`) or Tilda (`~`) symbol, the file name should contain at least one non-leading slash symbol. The string between the leading special symbol and that slash symbol is interpreted as an Account name, and the string after that slash symbol - as the name of the file to retrieve from the File Storage of the specified Account.

If the specified file name starts with the slash (`/`) symbol, but it does not contain any other slash symbols, the leading slash symbol is removed.

The TFTP module reads or writes the specified files on behalf of the `tftpuser` in the Main Domain. This makes it possible to retrieve files from any Account File Storage directory outside the `private` directories.

To allow TFTP clients to access `private` directories or to allow TFTP clients to store files, modify the target directory [File Access Rights](#), granting the `tftpuser` the Read and/or Write rights.

Examples:

| TFTP *filename* parameter | Addressed file |
|---|---|
| `file1.dat` | `file1.dat` in the Default File Storage |
| `/file1.dat` | `file1.dat` in the Default File Storage |
| `dirA/file1.dat` | `file1.dat` in the dirA subdirectory of the Default File Storage |

| | | |
|---|---|---|
| `/john/file1.dat`<br>`~john/file1.dat` | | `file1.dat` in the Account john File Storage |
| `/john/dirB/file1.dat`<br>`~john/dirB/file1.dat` | | `file1.dat` in the dirB subdirectory of the Account john File Storage |
| `/john@domain1.dom/dirB/file1.dat`<br>`~john@domain1.dom/dirB/file1.dat` | | `file1.dat` in the dirB subdirectory of the Account john@domain1.dom File Storage |

If the Try IP-Address Directory option is enabled, and the specified file name does not start with the slash or Tilda symbol, the module appends the text presentation of the client IP address in front of the file name. For a file read operation, if a file with this name is not found, the inserted prefix is removed, and the module re-tries to retrieve a file.

This feature allows you to create subdirectories inside the Default Storage directory, named with certain client IP addresses.

Examples:

| TFTP *filename* parameter | Client IP address | Addressed file |
|---|---|---|
| `file1.dat` | 10.0.1.0 | `10.0.1.0/file1.dat` (if absent when reading, use `file1.dat`) in the Default File Storage |
| `/file1.dat` | 10.0.1.0 | `10.0.1.0/file1.dat` (if absent when reading, use `file1.dat`) in the Default File Storage |
| `dirA/file1.dat` | 10.0.1.0 | `10.0.1.0/dirA/file1.dat` (if absent when reading, use `dirA/file1.dat`) in the Default File Storage |
| `/john/file1.dat`<br>`~john/file1.dat` | 10.0.1.0 | `file1.dat` in the Account john File Storage |

# ACAP Module

- **Application Configuration Access Protocol**
- **Configuring the ACAP module**

The CommuniGate Pro ACAP module implements an ACAP server for TCP/IP networks.

The ACAP protocol allows a client application to connect to the Server computer and upload and download the application preferences, configuration settings, personal address books, and other datasets.

# Application Configuration Access Protocol

The Application Configuration Access Protocol allows mailers and other application store any type of structured data on an ACAP server. That data can be the application configuration data, so when the application is started on any workstation, it can connect to the ACAP server and configure itself using the configuration stored in the ACAP 'dataset' belonging to the current user.

ACAP 'datasets' can also be used to store address books with those applications (like Mulberry®) that can work with address books stored on an ACAP server. The CommuniGate Pro WebUser Interface module uses the same datasets to store its address books. This feature allows CommuniGate Pro users to use the same personal address books via the WebUser Interface and via an ACAP-enabled mailer.

# Configuring the ACAP module

Use the WebAdmin Interface to configure the ACAP module. Open the Access page in the Settings realm:

| Processing | | |
|---|---|---|
| Log Level: Major & Failures | Channels: 10 | **Listener** |

Log

Use this setting to specify what kind of information the ACAP module should put in the Server Log. Usually you should use the Major (password modification reports) or Problems (non-fatal errors) levels. But when you experience problems

with the ACAP module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well.

The ACAP module records in the System Log are marked with the `ACAP` tag.

## channels

When you specify a non-zero value for the `TCP/IP` Channels setting, the ACAP module creates a so-called "listener" on the specified port. The module starts to accept all ACAP connections from mail clients and other applications. This setting is used to limit the number of simultaneous connections the ACAP module can accept. If there are too many incoming connections open, the module will reject new connections, and the user should retry later.

## listener

By default, the ACAP module Listener accepts clear text connections on the TCP port 674. Follow the listener link to tune the ACAP Listener.

The ACAP module supports the STARTTLS command that allows client applications to establish a connection in the clear text mode and then turn it into a secure connection.

---

# CommuniGate Pro

**Introduction**

**Installation**

**SysAdmin**

**Network**

**Objects**

**Storage**

**E-mail**

**Real-Time**

**Access**

**Services**

**Directory**

**Clusters**

**Applications**

**Miscellaneous**

**Licensing**

**WebMail**

**Pronto!**

**PBX**

| **Services** | HTTP | LDAP | PWD | RADIUS | SNMP | STUN | BSDLog |

# Services

- **Account Services**
- **Multi-Domain and Multihoming**

The CommuniGate Pro Server allows users to use various applications and protocols. This section describes the protocols that do not deal directly with E-mail transfer, real-time communication, or with access to Account data.

- The HTTP modules are HTTP servers provides access to the Server Administrator (WebAdmin) and to the WebUser Interfaces.
- The LDAP module is an LDAP server that provides access to various directories and databases.
- The PWD module is a poppwd server that allows users to change the account passwords using certain POP and IMAP mailers. It also implements the CLI/API interface.
- The RADIUS module is a RADIUS server that allows network access servers and other *edge devices* to authenticate CommuniGate Pro users.
- The SNMP module is an SNMP server ("agent") that can be used to monitor the CommuniGate Pro Server load and other statistical data.
- The STUN module is an STUN server that can help the CommuniGate Pro clients to deal with NAT traversal problems.
- The BSDLog module is an "BSD syslog" server that can be used to consolidate log records from 3rd party software into the the CommuniGate Pro Server Logs.

# Account Services

Every CommuniGate Pro Account can be used with Service modules. Several applications can use the same CommuniGate Pro Account at the same time, via the same, or different access and service modules.

# Multi-Domain and Multihoming

The Service modules support Multi-Domain and Multihoming configurations in the same way as the Access modules do.

**CommuniGate Pro**

| Services | HTTP | LDAP | PWD | RADIUS | SNMP | STUN | BSDLog |
|---|---|---|---|---|---|---|---|

# HTTP Modules

- **WebAdmin Server Interface**
- **WebAdmin Domain Interface**
- **WebUser Interface**
- **Access to Account File Storage**
- **Configuring the HTTP modules**
- **Routing**
- **Common Gateway Interface (CGI)**
- **Command Line Interface (CLI/API) Access**
- **HTTP Client**

The CommuniGate Pro HTTP Admin and User modules implement the Hypertext Transfer Protocol via TCP/IP networks.

The HTTP Admin module:

- provides access to the Server WebAdmin (Administration) Interface pages.
- provides access to Domain WebAdmin (Administration) Interface pages.

The HTTP User module:

- implements the WebUser Interface to user Accounts and Mailing List archives.
- implements the AirSync protocol.
- provides HTTP Binding for the XIMSS Interface.
- provides access to Account File Storage.
- provides access to the Parlay X interface.
- implements various WebDAV protocols (File Access, CalDAV, CardDAV, etc.) and Publish/Subscribe access methods to the Calendaring and Tasks data.
- provides access to Common Gateway Interface (CGI) programs.
- provides access to CG/PL Web applications.
- provides access to CLI/API.
- implements the Microsoft "Autodiscover" protocol.

The CommuniGate Pro HTTP modules support various authentication methods, including the GSSAPI, Kerberos, and Certificate methods implementing the *single sign-on* functionality.

The HTTP User module also implements an HTTP client. This functionality is used to send HTTP requests to and received HTTP responses from remote servers.

# WebAdmin Server Interface

The Server Administrator can use any Web browser application to configure and to monitor the Server remotely, using the Web (HTML) *forms*.

The authentication schemes supported with the HTTP protocol protect the WebAdmin pages from an unauthorized access. In order to access the WebAdmin pages, the user should provide the name and the password of a CommuniGate Pro Account with required [Server Access Rights](#).

By default, the HTTP Admin module accepts *clear text* TCP/IP WebAdmin connections on the port 8010 and *secure* (SSL/TLS) connections on the TCP port 9010.

To access the WebAdmin pages, the Server administrator should use the following URLs:
```
http://domain.com:8010
```

```
https://domain.com:9010
```
where *domain.com* is the CommuniGate Pro Main Domain name or its alias, or the IP address of the CommuniGate Pro Server.

Server configuration errors can cut you off the WebAdmin Server Interface, if all your Server IP addresses and DNS names are assigned to secondary Domains.

To access the WebAdmin Server Interface, use the following URLs:

```
http://sub.domain.com:8010/MainAdmin/
https://sub.domain.com:9010/MainAdmin/
```
where *sub.domain.com* is any name pointing to your Server computer or any of the Server IP addresses.

# WebAdmin Domain Interface

If the CommuniGate Pro Server supports several [Secondary Domains](#), the same port can be used by the [Domain Administrators](#) to access the Secondary Domains settings and account lists.

A Domain Administrator should access the server using the following URL:

```
http://sub.domain.com:8010
https://sub.domain.com:9010
```
where *sub.domain.com* is the name of the secondary Domain to administer.

The Server will ask for a user (Account) name and a password, and if the specified Account has the [Domain Administrator access right](#), the list of the Domain Accounts is displayed.

Sometimes this URL cannot be used. For example, a secondary Domain may have no DNS A-records (only MX records). To access such a Domain, its Domain Administrator should use the following URL:

```
http://domain.com:8010/Admin/sub.domain.com/
```
where:

> *domain.com* is the name of the Main Server Domain or its alias, or the IP address of the CommuniGate Pro Server.
> *sub.domain.com* is the name of the Domain to access.

Other Domains can specify your Domain as their [Administrator Domain](#). The Domain Settings page of your Domains provides a list of those Domains:

| Administrated Domains | |
|---|---|
| [node100.example.com](#) | settings |
| [test-dom.example.com](#) | settings |

You can open their WebAdmin Domain Interfaces using the links on this page. Remember that you should login using your full Account name (*yourAccountName@yourDomainName*) when accessing other Domain WebAdmin pages.

# Access to the WebUser Interface

CommuniGate Pro users can connect to the CommuniGate Pro Server with any Web browser (via the HTTP protocol) to manage their Accounts, to browse their Mailboxes, to read, copy, delete, forward, and redirect messages, to move messages between Mailboxes, to compose and submit new messages, etc.
This CommuniGate Pro component is called the WebUser Interface.

Registered users and guests can also use this component to browse Mailing List archives.

By default, the HTTP User module accepts *clear text* TCP/IP connections on the TCP port 8100, and the secure connections - on the TCP port 9100. If your Server does not have to coexist with some other Web Server on the same computer, it is recommended to change these port numbers to 80 and 443 - the standard HTTP and HTTPS port numbers.
In this case your users will not have to specify the port number in their browsers.

# Access to Account File Storage

CommuniGatePro users can user their Account File Storage as personal Web sites. See the File Storage section for the details.

The URL for the accountName@domainName File Storage (or personal Web site) is:

    http://domainName:port/~accountName

    https://domainName:port/~accountName

where `port` is the HTTP User ports (8100 and 9100 by default).

The list of files in that File Storage can be seen at:

    http://domainName:port/~accountName/index.wssp

    https://domainName:port/~accountName/index.wssp

This page can be used to manage the File Storage. It is available to the Account owner, Domain Administrators or, and to authenticated user granted the File Access Rights .

The "davmount" (RFC4709) document can be retrieved using the following URLs:

    http://domainName:port/~accountName/davmount

    https://domainName:port/~accountName/?davmount=1

You can specify an alternative File Storage prefix using the Domain Settings. That setting can be an empty string, in this case personal Web sites can be accessed using the following URLs:

    http://domainName:port/accountName/

    https://domainName:port/accountName/

You can also use the CommuniGate Pro Router to access personal Web sites with domain-level URLs. See the Routing section below.

# Configuring the HTTP modules

Use the WebAdmin Interface to configure the HTTP modules. Open the Services pages in the Settings realm, then open the HTTPA (Admin) and/or HTTPU (User) page.

| Processing | | | |
|---|---|---|---|
| Log Level: All Info | | Channels: 100 | **Listener** |
| Request Size Limit: 30M | | Scan Large Requests | |
| Compress If Larger: 10K | | Support 'Keep-Alive' | |

| HTTP Options | |
|---|---|
| Advertise 'Basic' Authentication | Advertise 'Digest' Authentication |
| Advertise 'NTLM' Authentication | Advertise 'Negotiate' Authentication |

Log Level

Use this setting to specify what kind of information the HTTP module should put in the Server Log. Usually you should use the `Major` or `Problems` (non-fatal errors) levels. But when you experience problems with the HTTP module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well.
The HTTP Admin module records in the System Log are marked with the `HTTPA` tag.
The HTTP User module records in the System Log are marked with the `HTTPU` tag.

Channels

This setting is used to limit the number of simultaneous TCP/IP connections the HTTP module can accept. Most browsers open several connections to load an HTML page and all embedded pictures, so do not set this limit to less than 5, otherwise some browsers may fail to download embedded graphic objects.

listener

Follow this link to open the HTTP Admin Port or HTTP User Port listener settings. There you can specify the TCP port number(s) the service should use, the interfaces to use, and other options.

If the CommuniGate Pro Server computer runs some other Web Server application, you should specify a port number in the "secondary range" to avoid conflicts with that other Web Server application. Usually the "secondary" Web Servers use ports numbers in the 8000-8100 range. If you set the port number to 8010, you will be able to connect to your server by entering `http://xxx.yyy.zzz:8010` in your Web browser, where *xxx.yyy.zzz* is the exact domain name (A-record) or the IP address of your Server.

Request Size Limit

This setting limits the maximum size of an HTTP request the Server accepts. You may want to increase this limit if you want to allow your users to upload larger files and/or to attach larger files to messages composed using he WebUser Interface.

Support Keep-Alive

If this option is enabled, the CommuniGate Pro supports the Keep-Alive protocol feature, so a user browser can retrieve several files using the same HTTP connection. Some browsers do not implement this function correctly and they may have problems if this option is enabled.

Compress If Larger

If the composed response size is larger than the value of this option, and the client browser supports compression, the response is compressed before being sent to the client.

Scan Large Requests

If this option is disabled, and the size of a HTTP request to be received exceeds the specified limit, the Server sends an error response and closes the connection. Many browsers do not display the error response in this case.

Instead, they display a generic "connection is broken" message.
If this option is enabled, the Server sends an error response back and receives the entire request (but it does not store it anywhere). The user browser then displays the error message.

`Advertise Basic AUTH`, `Advertise Digest AUTH`,
`Advertise NTLM AUTH`, `Advertise Negotiate AUTH`

If these options are enabled, the Server will inform browsers that clear-text (Basic) or secure (Digest, NTLM, Negotiate/SNEGO) authentication methods are available. If you plan to connect to the WebAdmin Interface via clear-text (non-encrypted) links over the Internet, you may want to enable these options. Different browsers work differently when these options are enabled. Some may fail, some may still use the clear-text (Basic) authentication method, so enable these options only if needed.
**Note:** the GSSAPI (Kerberos) authentication methods become available when the Negotiate AUTH method is enabled.

The HTTP modules set the MIME `Content-Type` for every object they send. To detect the proper Content-Type for plain files, the modules use the file name extension and the following "basic" built-in table:

| File Extension | MIME type |
|---|---|
| html | text/html |
| txt | text/plain |
| gif | image/gif |
| jpg | image/jpeg |
| css | text/css |
| js | text/javascript |

There is also an "extended" built-in table, which is displayed on the WebAdmin page.

You can create your own custom extension table by specifying additional file name extensions and corresponding MIME Content-Types:

| MIME types | |
|---|---|
| **File Extension** | **MIME type** |
| | |
| doc | application/msword |
| eml | message/rfc822 |
| flv | video/x-flv |
| htm | text/html |
| js | text/javascript |
| jss | text/javascript |
| mov | video/quicktime |
| mp3 | audio/mpeg |
| mpg | video/mpeg |
| pdf | application/pdf |

The extended "built-in" table is displayed under the custom table.

When a file extension is converted into a MIME type, the custom table is checked first, then the built-in tables are checked. As a result, you can redefine the built-in table values with the custom table values

.

# Routing

The HTTP modules use the [Router](#) to process all address it receives. But unlike other [Access](#) modules, the HTTP modules often deal not with complete E-mail addresses, but with domain names only.

When the HTTP Admin module receives a request, it uses the name or the IP address specified in the URL to decide which Domain Administration pages to display.

When the HTTP User module receives a request, it uses the name or the IP address specified in the URL to decide which Domain (its login page, Mailing Lists, File Storage, etc.) to access.

In order to support all types of CommuniGate Pro Routing features (Router Table, Domain Aliases, IP Address to Domain Mapping, etc.), the HTTP module composes a complete E-mail address `LoginPage@`*`domainname`* (where *domainname* is the domain name specified in the request URL), and then it processes this address with the Router:

- If the `LoginPage@`*`domainname`* address is routed to any module other than LOCAL, an error is returned.
- If the `LoginPage@`*`domainname`* address is routed to the LOCAL module and local part of the resulting address is still `LoginPage`, then the domain part of the resulting address is used to open the proper CommuniGate Pro Domain.
- If the address is routed to the LOCAL module, but the resulting username is not LoginPage, the [File Storage](#) of the addressed Account is opened. If the resulting address has a local part, then the File Storage subdirectory with that name is opened.

Samples ([Router](#) Table records, domainA.com is a CommuniGate Pro Domain):

`mail2.domain.com = domainA.com`
> When the `http://mail2.domain.com/` URL is used, the `<LoginPage@mail2.domain.com>` address is routed to `<LoginPage@domainA.com>` address, and the domainA.com Web Interface is opened.

`<LoginPage@mail2.domain.com> = user@domainA.com`
> When the `http://mail2.domain.com/` URL is used, the `<LoginPage@mail2.domain.com>` address is routed to LOCAL account user@domainA.com, and the user@domainA.com File Storage is opened.

`<LoginPage@mail2.domain.com> = subDir@user@domainA.com.domain`
> When the `http://mail2.domain.com/` URL is used, the `<LoginPage@mail2.domain.com>` address is routed to the LOCAL account user@domainA.com with the subDir local address part, and the subDir directory of the user@domainA.com File Storage is opened.

---

# Common Gateway Interface (CGI)

The HTTP User module can start CGI programs and scripts. To access a CGI program, the `/cgi-bin/`*`programName`* URL should be used, and the *programName* program should be placed into the CommuniGate Pro CGI Directory.

Use the WebAdmin Interface to open the HTTP User settings page:

| CGI Applications | | |
|---|---|---|
| CGI Directory: | | |
| **File Extension** | **Program to Start** | |
| | | |
| **HTTP Header Fields** | | |

CGI Directory
> Enter the name of the server system file directory where your CGI programs are located. If that directory is inside the CommuniGate Pro *base directory*, then you can specify its relative name, otherwise specify the full path to that directory. The CGI Directory should not have any subdirectories.

File Extensions
> While some operating systems (such as Unix) can start various interpreter-type programs/scripts by starting the proper interpreter, other operating systems require that the interpreter program is started explicitly. The CommuniGate Pro Server supports those operating systems by allowing you to specify the name of the interpreter program for certain file extensions. As a result, when the CommuniGate Pro Server needs to start a program or script with the specified extension, it forms the OS-level command line by prefixing the program/script name with the specified interpreter program name.
> In the above example the Server will process the `/cgi-bin/script.pl` URL by executing the

```
Perl.exe script.pl
```

> command.

HTTP Header Field
> Use this table to specify custom, non-standard HTTP Request header fields that you want to pass to your CGI applications. If a request contains a header line with the specified field name, the line is passed to CGI applications as an environment variable with the `HTTP_H_convertedFieldName` name, where convertedFieldName is the field name in the uppercase letters, with all minus (-) symbols substituted with the underscore (_) symbols.

CGI programs can be used to extend functionality of the WebUser Interface. They can log into the Server via its PWD module to do some CLI/API operations and/or via the IMAP or XIMSS modules to access and modify Mailbox data. To simplify these login operations, CGI programs can use the SessionID Authentication method.

---

# Command Line Interface (CLI/API) Access

The HTTP User module provides access to the Server Command Line Interface/API via the `/CLI/` realm.

## Text Method

Send a GET or POST request with a `command` parameter. The parameter value should contain the CLI command to be executed.
If the request fails, the error code is returned as an HTTP response error string.
If the request is successful, the HTTP response code is 200.
If the request produced an output, the text representation of the resulting object is sent as the HTTP response body.

## SOAP Method

Send a SOAP XML request. The request should have exactly one XML element - the CLI command to execute.
The XML element tag is the CLI command tag.
the XML element sub-elements are XML representations of parameter objects and/or `key` elements. The `key` element text body is a CLI command key.

Example:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <createAccount>
      <param>newuser@domain.dom</param>
      <param>TextMailbox</param>
      <param>
        <subKey key="RealName">John Doe</subKey>
        <subKey key="Password">soappass</subKey>
      </param>
    </createAccount>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

This request is converted into:

```
createAccount "newuser@domain.dom" TextMailbox {RealName="John Doe"; Password="soappass";}
```

If the request produces an output, its XML representation is added to the SOAP response.

Example:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <getAccount>
      <param>newuser@domain.dom</param>
    </getAccount>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

This request is converted:

```
getAccount "newuser@domain.dom"
```

and a dictionary output is produced:

```
{Password="\001xxxxx"; RealName="John Doe";}
```

This response is sent with the SOAP response:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <response>
      <object>
        <subKey key="Password">\001fjh{\024hdfu</subKey>
        <subKey key="RealName">SOAP Test</subKey>
      </object>
    </response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# HTTP Client

The HTTP Client functionality is used to send HTTP Requests to remote servers. It is used with:

- CG/PL `HTTPCall` function.
- XIMSS `retrieveURL` operation.

Use the WebAdmin Interface to specify the HTTP Client processing parameters. Open the General pages in the Settings realm, and find the HTTP Client Manager panel on the Others page:

| HTTP Client Manager | |
| --- | --- |
| Log Level: | All Info |
| Cache: | 30 |

Log
Use this setting to specify what kind of information the HTTP Client Manager should put in the Server Log. The HTTP Client Manager records in the System Log are marked with the `HTTPO` tag.

Cache
Use this setting to specify how many HTTP connections the Manager should keep open, increasing the processing speed for requests sent to the same remove HTTP server (these requests will reuse already open

connections).

The HTTP module sends a request to the specified `http:` or `https:` URL. The module accepts an optional parameter dictionary which can contain the following elements:

`method`
> If this element is specified, it specifies the HTTP request method (`GET`, `MOVE`, etc.)
> If this element is not specified, the HTTP request method is set to `GET` if the `body` element is absent, otherwise the `POST` method is used.

`urlParams`
> This optional element value should be a dictionary. All its key-value pairs are sent as URL parameters in the HTTP request line.
> Each dictionary value should be either a single value - a string or a number, or an array of single values. For an array-type value, a URL parameter for each array element is composed separately, using the same dictionary key.

`Content-Type`, `Content-Subtype`
> These optional elements specify the Content-Type header field for the HTTP request.

`body`
> If this optional element value is a dictionary:
>
> * the HTTP body is composed as "form data"
> * if the `method` element is specified, it must be `POST`
> * if the `Content-Type` element is not specified, the request `Content-Type` field is set to `multipart/form-data`.
> * if the `Content-Type` element is specified, it must be `multipart`, otherwise the `Content-Subtype` must be specified, too, and it must be set to `x-www-form-urlencoded`.
> * if the `Content-Type` element is not specified or if it is set to `multipart`, request body form data is composed using the `mutlipart/form-data` format, otherwise the `x-www-form-urlencoded` format is used.
> * each dictionary value should be be either:
>   * a single value - a string, a number, a datablock, a dictionary
>   * an array of single values
>
> For an array-type value, form data for each array element is composed separately, using the same dictionary key.
> If a single value is a dictionary, its empty-string ("") element value is used. If the request uses the `mutlipart/form-data` format, then the dictionary `Content-Type`, `Content-Subtype`, and (optionally) `charset` string element values are used to compose the form element `Content-Type` header field, and the `filename` string element (if it exists) is added to the form element `Content-Disposition` header field.
> Example:
> The following `body` element:
> ```
> {
>   field1 = "value1"; field2 = #145;
>   field3 = [YWJjYWJj]; field4 = ("value2",#777);
>   field5 = {"Content-Type"="image"; "Content-Subtype"="png"; "" =
> [shjhsjhkwuyuieyriuyuiyi];};
> }
> ```
> will be sent as
> ```
> --_____post-field_
> Content-Disposition: form-data; name="field1"
>
> value1
> --_____post-field_
> Content-Disposition: form-data; name="field2"
>
> 145
> --_____post-field_
> Content-Disposition: form-data; name="field3"
>
> abcabc
> --_____post-field_
> ```

```
Content-Disposition: form-data; name="field4"

value2
--_____post-field_
Content-Disposition: form-data; name="field4"

777
--_____post-field_
Content-Disposition: form-data; name="field5"
Content-Type: image/png

binary data
--_____post-field_--
```

If this optional element value is a string:

- the string is copied line-by-line into the request body
- the string EOL (End Of Line) separators are replaced with the Internet EOL (<carriage-return><line-feed>) symbols
- if the `Content-Type` element is not specified, the `Content-Type` field is set to `text/plain`.

If this optional element value is a datablock:

- the datablock is used as the request body
- if the `Content-Type` element is not specified, the `Content-Type` field is set to `application/octet-stream`.

If this optional element value is an XML Object:

- the XML Object text presentation is copied into the text body.
- if the `Content-Type` element is not specified, the `Content-Type` field is set to `text/xml`.

charset
    If this optional element is specified, it should be a name of a known character set. The request body is encoded from the UTF-8 encoding into the specified character set, and the `charset=charset_name` parameter is added to the `Content-Type` header field.
    If this element value is an empty string, no `charset=charset_name` header field parameter is added.
    If this element is not specified, the `charset=utf-8` header field parameter is added, but only if the Content Type of the request body is `text`.

closeConnection
    If this element is specified, the module adds the `Connection: close` header field to the request.

User-Agent
    If this element is specified, the module uses its string value as the custom `User-Agent` header field value. If this element value is an empty string, this header field is not sent at all.

Cookie
    This optional element value should be a string. It is used as the `Cookie` header field data.

If-Modified-Since
    The optional element should be either a timestamp or a string in the iCalendar date-time format. This element sets the HTTP request `If-Modified-Since` header field value.

responseFormat
    This optional element specifies how the HTTP response body should be converted (see below).

authName, authPassword
    These optional elements specify the credentials to send with the HTTP request.

timeout
> If this optional element is specified, it should be a number or a numeric string. Its value specifies the time-out for the HTTP request(s) sent. This value cannot exceed the maximum time-out value specified with the calling Server component.
> If this element is not specified, the maximum time-out value is used.

redirectLimit
> This optional element specifies how many time the HTTP module can resend the request to the new destination (URL) when it receives a 3xx response from a remote server
> If this element is not specified then the module resends a `GET` or `HEAD` up to 3 times, and does not resend other requests.

supplFields
> This optional element is a dictionary. The dictionary keys specified additional HTTP request field names, and the dictionary values (which should be strings or arrays of strings) specify the HTTP request field values.

The module returns either an error code, or a result dictionary with the following elements:

responseCode
> The numeric value of the HTTP response code (200 for successful operations, 300-399 for "redirect" responses the module has not processed itself, 400-599 for failed operations).

responseText
> A string containing the information text part of the HTTP response line.

Content-Type, Content-Subtype
> Strings containing the response body content type and subtype.

charset
> A string with the response body charset. If this element is present (it was specified as the `charset=charset_name` parameter of the HTTP response `Content-Type` header field, the response body is encoded from that character set into the UTF-8 encoding.

body
> This element exists if the HTTP response contains a non-empty body. This body is converted into the `body` response element according to the `responseFormat` request element value. If the responseFormat request element is not specified, the response Content type and subtype are used:

| responseFormat | Content-Type (if responseFormat is not specified) | converted to |
|---|---|---|
| xml | */xml, */*+xml | the body is interpreted as a text presentation of an XML Object. The `body` element is the XML Object read |
| calendar | */calendar | the body is interpreted as an iCalendar text. The `body` element is an [iCalendar XML Object](#) presenting the parsed body text. |
| *empty string* | */* | The `body` element is a datablock containing the HTTP response body. |

> If the module fails to convert the HTTP response body, then it returns an error if the responseFormat was specified. If the responseFormat was not specified, then the HTTP response body is returned as a datablock.

Date, Last-Modified, Expires
> Timestamp elements with the HTTP response header field values.

Server, Location

string elements with the response header field values.

`Set-Cookie`
    string or array of strings with the response header field value(s).

Services  HTTP  **LDAP**  PWD  RADIUS  SNMP  STUN  BSDLog

# LDAP Module

- **Lightweight Directory Access Protocol**
- **Configuring the LDAP module**
- **Client Authentication**
- **Central Directory**
- **Router Subtree**
- **LDAP Provisioning**
- **The `mail` Attribute processing**
- **Paging Search Results**

The CommuniGate Pro LDAP module implements an LDAP server for TCP/IP networks.

The LDAP protocol allows a client application (a mailer or a search agent) to connect to the Server computer and retrieve information from the server Directory. The LDAP protocol can also be used to modify data in the Directory.

Besides the Directory, the LDAP module provides access to the Account Manager for provisioning, and to the Router component for external filtering and other systems (E-mail address verifications).

The CommuniGate Pro LDAP module supports both *clear text* and *secure* SSL/TLS connections. The LDAP module supports the "Start TLS" command (RFC2830) that allows client applications to establish a connection in the clear text mode and then turn it into a secure connection.

# Lightweight Directory Access Protocol

The CommuniGate Pro LDAP module provides access to the CommuniGate Pro Directory tree and its records.

It is important to understand that the CommuniGate Pro LDAP module itself does not provide any Directory services. It just implements an access protocol, and the functionality it provides depends on the CommuniGate Pro Directory Manager and its units.

Very often LDAP services are used to look for names and E-mail addresses of Server users. But since the LDAP module provides access to the entire Directory tree, it can be used to work with any type of data placed into the CommuniGate Pro Directory. While the CommuniGate Pro Directory can be stored in several Storage Units - both local and remote, the LDAP clients see the entire Directory as one large tree.

To browse and modify the Directory, system administrators can use either LDAP clients and utilities, or the Directory Browser interface built into the CommuniGate Pro WebAdmin Interface.

**Note:** while the LDAP module implements an LDAP server functionality, the CommuniGate Pro Server can also work as an LDAP client, using the LDAP protocol to access external LDAP servers and their databases. Those external Directories are presented as subtrees of the CommuniGate Pro Directory tree. See the Remote Units Directory section for more details.

# Configuring the LDAP module

Use the WebAdmin Interface to configure the LDAP module. Open the Services pages in the Settings realm, and open the LDAP page.

| Processing | | |
|---|---|---|
| Log Level:    Major & Failures | Channels:    25 | **Listener** |

`Log`

Use this setting to specify what kind of information the LDAP module should put in the Server Log. Usually you should use the `Major` or `Problems` (non-fatal errors) levels. But when you experience problems with the LDAP module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well.

The LDAP module records in the System Log are marked with the `LDAP` tag. Please note that LDAP is a binary protocol, so all low-level data is presented in the hexadecimal form.

`Channels`

When you specify a non-zero value for the `TCP/IP` Channels setting, the LDAP module creates a so-called "listener" on the specified port. The module starts to accept all LDAP connections that mail clients establish in order to update password data. This setting is used to limit the number of simultaneous connections the LDAP module can accept. If there are too many incoming connections open, the module will reject new connections, and the user should retry later.
If the number of channels is set to zero, the LDAP module closes the listener and releases (unbinds from) the TCP port(s).

`listener`

By default, the LDAP module Listener accepts clear text connections on the TCP port 389, and secure connections - on the TCP port 636. Follow the listener link to tune the LDAP Listener.

**Note:** The pre-4.7 Netscape ® LDAP clients crash if they communicate with a very fast server returning more than 90 records. Ask your users to update to the 4.7 or later version of Netscape browser/mailer product.

**Note:** The Netscape® LDAP client (version 4.7) does not correctly process the "properties" command - it always tries to connect to the port 389, even if the search was successfully made on a different (for example, secure) port.

Sometimes you need to specify the Directory Tree Root element (an empty string) as the "search base DN". Some LDAP clients do not process this situation correctly (for example, Microsoft LDAP client silently replaces an empty Search Base string with the `c=your_country` string).
In these cases you should specify the string `top` as your Search Base string. The LDAP module interpretes this string as an empty string (Directory Root DN).

# Client Authentication

The Directory Access Rights are based on the so-called Bind DNs rather than on CommuniGate Pro Account names and Account rights. See the Directory Manager Access Rights section for more details.

The Directory Access Rights set by default do not require Directory (LDAP) clients to authenticate in order to retrieve any information from the Directory tree.

When an LDAP client tries to authenticate as a certain DN, the LDAP server retrieves the Directory record with the specified DN and compares that record `userPassword` attribute with the password supplied by the LDAP client. If the record exists, and it contains the `userPassword` attribute, and the attribute value matches the supplied password, the LDAP client authentication succeeds.

The LDAP module provides an alternative authentication method, when the client specifies a CommuniGate Pro Account name instead of some record DN. In this case, the CommuniGate Pro Server opens the specified Account and compares the Account password with the supplied password. If the passwords match, the Server builds a DN for the Account record using the Directory Integration settings, and uses it as the Bind DN.

Sample:

> If the Directory Integration settings are:
> Base DN:              `o=myCompany`
> Domain RDN attribute:  `cn`
>
> and the client has submitted the `user@domain.dom` name and the correct password for the user@domain.com account, then the LDAP client is authenticated with the following Bind DN:
> `uid=user,cn=domain.dom,o=myCompany`
> and this client can access the Directory information available for that Bind DN.

The LDAP module uses the alternative authentication method if the specified string does not contain any equals (`=`) symbol, or if it starts with the `mail=` symbols and does not contain any other equals (`=`) symbols.

This authentication service can be disabled by disabling the `LDAP` Service for a Domain and/or an Account.

The LDAP Provisioning option can modify the authentication process. If this option is enabled and the supplied Bind DN represents the DN for some CommuniGate Pro Account, the supplied Bind DN is converted into that Account name, and the alternative method is used.

| Bind DN string specified | Data used to verify the password |
|---|---|
| `uid=user,cn=domain.dom,o=myCompany` (LDAP Provisioning is off) | the `userPassword` record attribute of the `uid=user,cn=domain.dom,o=myCompany` Directory record |
| `ou=human_resources,o=myCompany` | the `userPassword` record attribute of the `ou=human_resources,o=myCompany` Directory record |
| `user@domain.com` | the `user@domain.dom` Account password |
| `mail=user@domain.com` | the `user@domain.dom` Account password |
| `uid=user,cn=domain.dom,o=myCompany` (LDAP Provisioning is on) | the `user@domain.dom` Account password |

The LDAP module allows users to employ all authentication methods supported with the CommuniGate Pro Server. It supports Simple, SASL, and NTLM BIND methods.

If the Account password authentication method is used, and the specified Account has the Directory Administrator access right, the LDAP client can access and modify all Directory data ("master"-type access).

# Central (users) Directory

The LDAP services can be used to retrieve information about the CommuniGate Pro Accounts and other Domain Objects.

To search the Directory for CommuniGate Pro Domain Objects (Accounts, Groups, Mailing Lists), the LDAP clients should be tuned to point to the proper Subtree (this parameter is called "Search Base" in many LDAP clients). The Directory Subtree for the `company.com` Domain is `cn=company.com,o=MyCompany`, where cn is the Domain RDN attribute, and o=MyCompany is the Base DN for CommuniGate Pro Domains. The Base DN and Domain RDN attribute are the [Directory Integration](#) settings and can be modified. If these settings are modified, the locations of Domain subtrees are changed, and the LDAP clients should be reconfigured to specify the new locations in their "Search Base" settings.

# Router Subtree

Some external application and devices (such as external E-mail filters) use the LDAP protocol to verify if the E-mail recipient should be accepted on behalf of the "main" mail server.
These applications check a *object@domain* E-mail address using one of the following requests:

- a 'record retrieval' (scope=base) request for the `mail=object@domain,searchbase` DN, or
- a 'record search' (scope=one) request for the `searchbase` DN, using the `mail=object@domain` filter

To support these application and devices, the CommuniGate Pro LDAP module implements a virtual `dc=cgprouter` Directory subtree. If a 'record retrieval' (scope=base) request DN is `mail=object@domain,dc=cgprouter`, or a 'record search' (scope=one) request DN is `dc=cgprouter`, and the search filter is `mail=object@domain`, the LDAP module does not consult the Directory. Instead, it tries to process the specified *object@domain* E-mail address using the [Router](#) component.

If the specified E-mail address is successfully routed to a local Account, or to the "Black Hole", or it is routed to an external host, but relaying to that host is allowed, the LDAP module returns a fictitious Directory record, informing an application that the supplied E-mail address is a valid one, and E-mail sent to that address can be accepted.

If the address can not be routed (or relaying is not allowed) and the request was the 'record search' one, the result is empty. If the request was the 'record retrieval' one, the result delivers routing error.

# LDAP Provisioning

CommuniGate Pro supports Directory-based Domains. Account information in those Domains is stored in the Directory, and the LDAP module can be used to modify Account data in the Directory. Directory-based Domains resemble the architecture of some older Messaging Systems, and CommuniGate Pro supports it for compatibility and migration reasons. See the [Directory-Based Domains](#) section for more details.

The LDAP module can also be used to perform basic provisioning operations within regular Domains. This feature is called LDAP Provisioning. If the DN specified in the request "looks like" a DN of a Directory record that some CommuniGate Pro Account has (or could have), the LDAP module does not perform any operation on the Directory at all. Instead of passing the request to the Directory, the LDAP module sends a command directly to the Account Manager, and the Account Manager creates/removes/renames/updates/reads the specified Account. See the [Directory Integration](#) section for more details.

# The `mail` Attribute processing

Many LDAP clients expect to see the `mail` attribute in Account and other Domain Object records. But, by default, CommuniGate Pro does not store such an attribute in those directory records.

If the LDAP module has to return such a record (a record of the CommuniGateAccount, CommuniGateMailList, or CommuniGateGroup object class), and that record does not contain the `mail` attribute, the LDAP module can compose that attribute on-the-fly, using the Object record DN: it takes the `uid` value from the DN (Account/Object name), the `cn` attribute value (Domain name), and merges them using the `@` symbol to build the uidValue@cnValue `mail` attribute value. As a result, when an object is renamed (its record `uid` attribute is changed), or when the Domain is renamed (the `cn` attribute in the object DN is changed), the `mail` attribute is automatically updated.

Since the `mail` attribute is not stored in the Directory records by default, all search filters that use the `mail` attribute can be modified internally to use the `uid` attribute instead. If the search operation is "equals to", and the search string contains the `@` symbol, only the part of the string before that symbol is used.

These two features can be enabled or disabled using the Domain Integration page in the Users realm of the WebAdmin Interface:

| LDAP Attribute Processing |
| --- |
| Substitute 'mail' with 'uid' in conditions |
| Compose 'mail' using 'uid' |
| Ignore 'objectCategory' conditions |

Ignore objectCategory filters
> Some clients (including Microsoft Outlook) always send their LDAP search requests using filters checking for certain `objectCategory` attribute values.
> Since this attribute is not included into CommuniGate Pro Account records by default, you would have to create this attribute as a Custom one, and put a "proper" value into each Account settings.
> This option tells the LDAP server to ignore all "objectCategory equals *value*" search operations (the Server treats these operations as the constant `true`-values).

The LDAP module checks if a search request explicitly specifies the `displayName` attribute. If a retrieved record does not contain that attribute, but the record contains the `cn` attribute, or the `uid` attribute, the value of the `cn` or `uid` attribute is included into the response as the `displayName` attribute value.

---

# Paging Search Results

The LDAP module supports the Extension for Simple Paged Results Manipulation ([RFC2696](#)) which allows LDAP clients to retrieve large search results in smaller portions (pages).
However, to return one page to a client the Server has to find all records matching the search criteria, then optionally sort them; and only then it can return the requested page. In order to prevent the performance degradation caused by repetitive searches and sortings the Server may cache several such search results using its internal memory.

The caching parameters can be adjusted using the Domain Integration page in the Users realm of the WebAdmin Interface:

| LDAP paged Search Results Cache | |
| --- | --- |
| Size: 0 | Used: 1 |
| Time-out: 4 min | |

Size
> The maximum number of cached search results.

Used
> The number of currntly cached results.

`Time-out`

The time period since the last client retrieval of a subsequent page, after which the memory used by cached data will be released.

# CommuniGate Pro

| Services | HTTP | LDAP | **PWD** | RADIUS | SNMP | STUN | BSDLog |
|----------|------|------|---------|--------|------|------|--------|

# PWD Module

- **Password Modification Protocol (poppwd)**
- **Configuring the PWD module**
- **Providing Access to the Server CLI**

The CommuniGate PWD module implements a poppwd server for TCP/IP networks.

The poppwd protocol allows a client application to connect to the Server computer and to specify the user (Account) name and the password. If access to the specified user account is granted, the mailer application sends the new password to the Server, and the server update the user password in the user account information data.

The PWD module also provides access to the Server Command Line Interface (CLI)

# Password Modification Protocol (poppwd)

The PWD module can be used to modify the CommuniGate Pro Account password. If the "old" password specified by a mail client matches the password set in the user's Account Settings, the new password is stored in the Account Settings.

The PWD module checks the Can Modify Password Account Settings option and refuses to modify an Account password if this option is disabled.

The PWD module supports the *clear text* authentication method, and it also supports the secure APOP and SASL AUTH authentication methods.

When used in a Cluster environment, the PWD module can update passwords on all Cluster member servers.

---

# Configuring the PWD module

Use a Web browser to open the Settings realm of the WebAdmin Interface. Open the Services pages, then open the PWD page.

| Processing | | |
|---|---|---|
| Log Level:  Major & Failures | Channels: | **Listener** |

Log

Use this setting to specify what kind of information the PWD module should put in the Server Log. Usually you should use the `Major` (password modification reports) or `Problems` (non-fatal errors) levels. But when you experience problems with the PWD module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well.
The poppwd clients send passwords in the clear text format, and setting the Log setting to these values for long periods of time can become a security hole, if the Log file can be copied from the Server computer.

The PWD module records in the System Log are marked with the `PWD` tag.

channels

When you specify a non-zero value for the `TCP/IP` Channels setting, the PWD module creates a so-called "listener" on the specified port. The module starts to accept incoming PWD connections.
This setting is used to limit the number of simultaneous connections the PWD module can accept. If there are too many incoming connections open, the module will reject new connections, and the user should retry later. If the number of channels is set to zero, the PWD module closes the listener and releases (unbinds from) the TCP port.

listener

By default, the PWD module Listener accepts clear text connections on the TCP port 106. Follow the listener link to tune the PWD Listener.
**Note:** Some versions of Apple MacOSX use the port 106 for Apple's own version of a Password Server. To avoid conflicts with that program, the default CommuniGate Pro PWD port on that OS is set to 8106.

# Providing Access to the Server CLI

As soon as a PWD user is authenticated, the Server Command Line Interface (CLI) commands are accepted. See the Command Line Interface section for the details.

| Introduction |
|---|
| Installation |
| SysAdmin |
| Network |
| Objects |
| Storage |
| E-mail |
| Real-Time |
| Access |
| **Services** |
| Directory |
| Clusters |
| Applications |
| Miscellaneous |
| Licensing |
|  |
| WebMail |
| Pronto! |
| PBX |

| Services | HTTP | LDAP | PWD | **RADIUS** | SNMP | STUN | BSDLog |
|---|---|---|---|---|---|---|---|

# RADIUS Module

- **Configuring the RADIUS Module**
- **RADIUS Authentication**
- **External Helper**
- **Accounting Log**

The CommuniGate Pro Server supports the RADIUS authentication and account protocol. It can be used with for various NAS (Network Access Server) devices and programs.

The RADIUS module acts as a RADIUS server. It receives authentication requests from RADIUS clients (NAS), verifies the supplied credentials and accepts or rejects these requests.

The RADIUS module supports the following authentication methods:

- PAP
- CHAP
- MS-CHAPv1
- MS-CHAPv2
- EAP
    - MD5-Challenge
- DIGEST-MD5

The RADIUS module can use an external helper application to implement site-specific access policy (based on RADIUS request attributes) and to return additional attributes to NAS.

# Configuring the RADIUS Module

By default the CommuniGate Pro RADIUS module is not activated.

CG/PL applications can communicate with remote RADIUS servers: they can send RADIUS requests and receive RADIUS responses. To enable this RADIUS client functionality, the RADIUS module has to be activated.

Use the WebAdmin Interface to configure the RADIUS module. Open the Services pages in the Settings realm, and open the RADIUS page:

| Processing | | |
|---|---|---|
| Log Level: Problems | | **Listener** |
| Password: | Require NAS ID | |
| Channels: 3 | Record | |

Log

Use this setting to specify what kind of information the RADIUS module should put in the Server Log. Usually you should use the `Major` or `Problems` (non-fatal errors) levels. But when you experience problems with the RADIUS module, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well.

The RADIUS module Log records are marked with the `RADIUS` tag. Please note that RADIUS is a binary protocol, so all low-level data is presented in the hexadecimal form.

listener
Use this link to open the UDP Listener page and specify the port number and local network address for the RADIUS server authentication service, and access restrictions for that port. When the port number is set to 0, the RADIUS server is disabled.
By default RADIUS clients send requests to the UDP port 1812.
If your server computer is already running some RADIUS server, you may want to specify a non-standard port number here and reconfigure your RADIUS client software to use that port number.

Channels
Use this setting to specify the number of RADIUS module processors (threads) used to process RADIUS requests. If you set this setting to 0, all requests will be processed directly with the RADIUS Listener thread(s).

Require NAS ID
The RADIUS protocol requires all requests to contain a `NAS-Identifier` or a `NAS-IP-Address` attribute (or both). If this option is not selected, requests without these attribute are accepted, and the word `unknown` is used as the Identifier in the module log records.

Password
Use this setting to specify the RADIUS "shared secret". All RADIUS clients should use the same "shared secret" in order to access the RADIUS server.

Record
If this option is enabled, the RADIUS module stores all Accounting requests in a text file. See the Accounting Log section below.

---

# RADIUS Authentication

The RADIUS module accepts properly formatted "Access-Request" requests from RADIUS clients, retrieves the User-Name and User-Password attributes and tries to find the specified CommuniGate Pro Account and verify its password. If the password can be verified and the Account and its Domain both have the RADIUS Service enabled, a positive response is sent to the RADIUS client, otherwise a negative response with the error code text is sent.

If the CommuniGate Password option is enabled for the specified Account, the RADIUS module checks if the Account has the `RADIUSPassword` setting. If it exists, it is used instead of the standard `Password` setting. This feature allows an Administrator to assign a alternative Account password to be used for the RADIUS authentication only.

**Note:** clients authenticating via RADIUS do not use any network address on the Server, and Secondary Domain users should specify their full Account name (*account@domain*), or should specify a name that is routed to their Account using the Router. Because the Router is used to process the User-Name attribute, account aliases can be used for authentication, too. See the Access section for more details.

---

# External Helper

The CommuniGate Pro Server can use an external Helper program to implement a RADIUS authentication policy. That program should be created by your own technical staff.

The program name and its optional parameters should be specified using the WebAdmin Helpers page. Open the General page in the Settings realm, and click the Helpers link:

| External RADIUS | |
|---|---|
| Enabled | |
| Log Level: Major & Failures | Program Path: |
| Time-out: disabled | Auto-Restart: 15 seconds |

See the Helper Applications section to learn about these options. The External RADIUS module System Log records are marked with the EXTRADIUS tag.

If the External RADIUS program is not enabled, then the positive authentication response is sent as soon as the user password is verified. The response does not contain any additional attributes.

To learn how to create your own External RADIUS programs, see the Helper Applications section.

Sample External RADIUS programs and scripts can be found at the RADIUS Helper programs site.

---

# Accounting Log

If the Record option is enabled, all RADIUS accounting operations are recorded in a text-based Accounting Log file. The Accounting Log files are stored inside the RADIUSLog file subdirectory.

A single-server system creates the RADIUSLog directory inside the Settings subdirectory of the *base directory*. A Dynamic Cluster system creates the RADIUSLog directory inside the Settings subdirectory of the *SharedDomains* directory.

Each RADIUS Accounting Log file has a *yyyy-mm-dd* file name (where *yyyy* is the current year, *mm* is the current month, and *dd* is the current month day), with the log file name extension. At local midnight, a new Accounting Log file is created.

Each RADIUS Accounting Log record is a text line containing a time-stamp, the operation type or command (started, ended, updated, inited, stopped), and optionally an account name.
The rest of the line contains accounting request attributes.
Each attribute is stored using the numeric attribute type, the equal (=) symbol, and the attribute value.
Attribute values are encoded in the same way as in they are encoded in dictionaries used in External RADIUS Helper Interface.

---

# CommuniGate Pro

| Services | HTTP | LDAP | PWD | RADIUS | **SNMP** | STUN | BSDLog |

## SNMP Module

- **Configuring the SNMP Module**
- **Accessing the Server MIB**
- **Sending SNMP Traps**

The CommuniGate Pro Server maintains a set of [Statistics Elements](#) containing information about the Server activity. These Elements can be be accessed via the built-in SNMP server ("agent").

The SNMP agent receives requests from SNMP clients ("managers") and either returns the information about internal states, counters, problems, or modifies the internal settings by a client request.

The [Setting up MRTG for CommuniGate Pro](#) document should help you configure the popular freeware SNMP manager.

## Configuring the SNMP Module

By default, the CommuniGate Pro SNMP agent is not activated.

Use the WebAdmin Interface to configure the SNMP Module. Open the Services pages in the Settings realm, and open the SNMP page:

| Processing | | |
|---|---|---|
| Log Level: | Problems | **Listener** |
| Password: | | |
| Trap Password: | | |
| Trap Protocol: | SNMPv1 | |

`Log`

Use this setting to specify what kind of information the SNMP agent should put in the Server Log. Usually you should use the `Major` or `Problems` (non-fatal errors) levels. But when you experience problems with the SNMP agent, you may want to set the Log Level setting to `Low-Level` or `All Info`: in this case protocol-level or link-level details will be recorded in the System Log as well.

The SNMP agent records in the System Log are marked with the SNMP tag. Please note that SNMP is a binary protocol, so all low-level data is presented in the hexadecimal form.

listener
Use this link to open the UDP Listener page and specify the port number and local network address for the SNMP agent, and access restrictions for that port. When the port number is set to 0, the SNMP agent is disabled. By default SNMP Manager programs send requests to the UDP port 161.
If your server computer is already running some other SNMP agent, you may want to specify a non-standard port number here and reconfigure your SNMP Manager software to use that port number.

Password
Use this setting to specify the SNMP "community name". The CommuniGate Pro SNMP agent accepts only those SNMP requests that contain the proper "community name" data.

Trap Password
This setting specifies an alternative SNMP "community name". The CommuniGate Pro SNMP agent accepts SNMP requests containing this "community name", and remembers the network (IP) addresses those requests have come from. The module then can send SNMP Traps to all remembered addresses.

Trap Protocol
This setting specifies the SNMP protocol version to use when sending SNMP Traps.

# Accessing the Server MIB

The MIB (Management Information Base) is a text file describing the internal objects the SNMP agent can display, monitor, and/or modify. You need the CommuniGate Pro MIB file to properly configure the SNMP client ("manager") you want to use for server monitoring.

Different versions of the CommuniGate Pro software support different sets of internal objects, and CommuniGate Pro Server generates its MIB by a user request, presenting the most current information.

To access the CommuniGate Pro MIB file, use the WebAdmin interface to access the CGatePro-MIB.txt file:
`http://yourservername:8010/CGatePro-MIB.txt`

Save this file to a monitoring workstation disk and use it to configure your SNMP manager software.

# Sending SNMP Traps

The SNMP module can send Traps on certain Events. See the Statistics Triggers section for more details. Traps can be sent to the network addresses explicitly specified in the Event Handler settings and/or to all remembered network addresses - addresses from which SNMP requests with the "Trap Password" *community name* have been received.

The SNMP module sends Traps using the SNMP protocol version specified with the Trap Protocol setting.

| Services | HTTP | LDAP | PWD | RADIUS | SNMP | STUN | BSDLog |
|---|---|---|---|---|---|---|---|

# STUN Module

- **Configuring the STUN Module**
- **Protocol Multiplexing**

The CommuniGate Pro Server supports the STUN protocol.

The STUN module acts as a STUN server. It receives requests from client systems and sends back responses, containing the IP Network information that these systems can use to detect the type of NAT network they are in.

# Configuring the STUN Module

Use the WebAdmin Interface to configure the STUN module. Open the Services pages in the Settings realm, and open the STUN page:

| Processing | | |
|---|---|---|
| Log Level:   Problems | **UDP Listener** | |
| | **TCP Listener** | Channels:   30 |

Log

Use this setting to specify which records the STUN module should put in the Server Log.

The STUN module Log records are marked with the STUN tag.

Listener

Use this link to open the UDP Listener and TCP Listener pages and specify the port numbers and local network addresses for the STUN service, and access restrictions for these port.

By default STUN clients send "Bind" requests to the UDP port 3478, and they send "Shared Secret" requests to the secured TCP (TLS) port 5349.

```
Channels
```
   Use this setting to specify the maximum concurrent number of STUN TCP requests.


**Note:** the "old" STUN protocol (RFC 3489) requires 4 UDP sockets to be configured, using 2 different ports and 2 different IP Addresses, in all combinations. To support the RFC 3489 protocol, the STUN server should have at least 2 different Local IP Addresses available to its clients.

# Protocol Multiplexing

The [SIP](#) module supports STUN packet multiplexing with the SIP protocol. When a STUN packet is received on any of the SIP UDP Listener sockets, the packet is passed to the STUN module for processing.

**Note:** the STUN module must know the exact Network IP Address the request was received on. To support STUN protocol, the SIP Module UDP Listener sockets must be configured using explicitly specified IP Addresses, and not use the "all addresses" settings.

**Note:** the SIP module UDP Listener sockets can be used as "alternative" STUN address and port. I.e. you can create only 2 UDP Listener sockets for the network address IP1, port 3478, and for the network address IP2, port 3478 - if there are at least 2 SIP UDP Listener sockets, for:

- network address IP1, port 5060 (or any port other than 3478)
- network address IP2, port 5060 (or any port other than 3478)

To support ICE clients, the STUN module supports STUN packet multiplexing with the RTP protocol: the Media Proxy and Media Server legs detect STUN packets and send them the the STUN module for processing.

# CommuniGate Pro

| Services | HTTP | LDAP | PWD | RADIUS | SNMP | STUN | **BSDLog** |

## BSDLog Module

- **Configuring the BSDLog Module**

The CommuniGate Pro Server supports the BSD syslog protocol.

The BSDLog module acts as a BSD syslog server. It receives syslog requests from other systems, and stores the supplied syslog records in the CommuniGate Pro Logs, and, optionally in the Server OS syslog.

By default the CommuniGate Pro BSDLog syslog server is not activated.

---

Sidebar navigation:

Introduction
Installation
SysAdmin
Network
Objects
Storage
E-mail
Real-Time
Access
Services
Directory
Clusters
Applications
Miscellaneous
Licensing

WebMail
Pronto!
PBX

---

# Configuring the BSDLog Module

Use the WebAdmin Interface to configure the BSDLog module. Open the Services pages in the Settings realm, and open the BSDLog page:

| Processing | | |
|---|---|---|
| Log Level: Problems | | **Listener** |
| Use OS 'syslog' | | |

Log

Use this setting to specify which records the BSDLog module should put in the Server Log. The BSDLog module assigns:

- the Crash level to syslog messages with Severity Codes 0,1, and 2
- the Failure level to syslog messages with Severity Code 3
- the Major level to syslog messages with Severity Code 5
- the Problem level to syslog messages with Severity Code 4
- the Low-Level level to syslog messages with Severity Code 6
- the All Info level to syslog messages with Severity Code 7

The BSDLog module Log records are marked with the `BSDLog` tag.

`listener`
Use this link to open the UDP Listener page and specify the port number and local network address for the BSDLog service, and access restrictions for that port. When the port number is set to 0, the BSDLog server is disabled.

By default syslog clients send requests to the UDP port 514.

If your server computer is already running some "BSD syslog" server, you may want to specify a non-standard port number here and reconfigure your syslog client software to use that port number.

`Use OS syslog`
If this option is enabled, the BSDLog module stores all received records via the Server OS syslog service.

---

| Directory | Schema | Integration |
|-----------|--------|-------------|

# Directory

- **What is Directory?**
- **Directory Storage Units**
- **Local Storage Units**
- **Remote Storage Units**
- **Remote Directory Root**
- **Binding to the Directory**
- **Access Right Records**
- **Access Right Specifications**
- **Directory Browser**
- **Importing Directory Data**
- **Cluster-wide Directory Units**

The CommuniGate Pro Server includes the Directory Manager implementing high-performance standards-based Directory storage.

The Directory can contain records about the CommuniGate Pro Accounts, Domains, and other objects. It can also contain any other type of records, and it can be used as a stand-alone Directory Server serving any LDAP-based applications.

The Directory Manager is not the same as an LDAP server. The CommuniGate Pro LDAP module provides access to the Directory for LDAP clients, but various CommuniGate Pro components (Account and Domain Managers, WebUser Interface, etc.) access the Directory data directly, bypassing LDAP communications.

The Directory Manager implements *Meta-Directory*: it can store directory data in one or several sets of the server files (Local Units), and it can also use external LDAP servers as Directory Remote Units. Many different configurations are possible. The following simplest configurations are used most often:

- All Directory data is stored in a single Local Unit. In this case Meta-Directory is the same as a Directory implemented with a regular LDAP server.
- All Directory data is stored in a single Remote Unit. In this case Meta-Directory is used just as a method to access records stored on an external LDAP server.

# What is Directory?

*attribute*

a name (*attribute name*) and one or several *attribute values*.
Usually an attribute is presented in the `name=value` form.
Attribute names are case-insensitive.
Samples:

```
userName=john
eyeColor=blue
```

*object class*

an attribute with the `objectClass` name; this attribute is used to specify a nature of the object it belongs to.
Samples:

```
objectClass=person
objectClass=organization
```

*distinguished name* (DN).

a sequence of *attributes* presented in the `name=value` form and separated with the comma (`,`) symbol.
DNs are used as unique names for objects (records).
Sample:

```
userName=john,server=BigIron,realm=Internet
```

DNs are used to build object name trees, with the rightmost attribute specifying the most generic name, and the leftmost attribute specifying the unique object name itself.
The leftmost attribute is called *Relative Distinguished Name* (RDN) - it provides a unique name for the object among all objects with DNs having the same parent DN.
Sample:

```
userName=jim,server=BigIron,realm=Internet
```
this is a different DN, but it has the same "parent DN" (`server=BigIron,realm=Internet`)

Sample:

```
userName=john,server=SmallCopper,realm=Internet
```
this is a different DN, with a different "parent DN" (`server=SmallCopper,realm=Internet`)

*directory record* or *object*

set of *attributes* with a *distinguished name*
Usually a record is presented as several lines starting with the name presenting the record DN, followed by the lines presenting the record attributes. Several records are usually separated with an empty line.
Sample:

```
DN: userName=jim,server=BigIron,realm=Internet
objectClass=person
eyeColor=blue
mailboxLimit=1024000

DN: userName=john,server=BigIron,realm=Internet
objectClass=person
eyeColor=green
mailboxLimit=2048000
```

Note: the LDAP standard recommends to include the RDN attribute into the set of attributes making up a directory record. CommuniGate Pro Directory Manager enforces this rule.

*directory*

a set of *directory records*; this can be a very large set (millions of records). The set is organized as a tree using

DNs. Records are removed automatically when the record with the parent DN is removed. Record DNs are updated automatically when the parent DN is changed (renamed).

*directory schema*
a set of directory restrictions, including:

- a set of *attribute names* that can be used in the Directory (`userName, mail, city, eyeColor, ...`);
- a set of objectClass attribute values that can be used in the Directory (`person, organization, device, printer ...`);
- for each objectClass - names of the attributes that must be present in the object record; for records with `objectClass=person` a schema may require attributes with `cn` (canonical name) and `sn` (surname) names;
- for each objectClass - names of the attributes that may be present in the object record; for records with `objectClass=person` a schema may allow attributes with `driverLicense` and `eyeColor` names.

# Directory Storage Units

While the entire CommuniGate Pro Directory is presented to its clients as one large tree of directory records, its subtrees can be stored in separate *storage units*. This type of "virtual" directories is often called Meta-Directory.

CommuniGate Pro Directory supports two types of storage units:

- *local* units - sets of files managed with the CommuniGate Pro File Directory Manager. These files contain directory records, replication information, and subtree schemas.
- *remote* units - descriptors managed with the CommuniGate Pro LDAP Directory Manager. These descriptors contain the information about remote Directories accessed via LDAP.

As a result, the CommuniGate Pro Directory may include subtrees located on remote servers. If an LDAP server ldap.server.dom provides access to some directory tree, you can create a remote unit in the CommuniGate Pro Directory that points to the ldap.server.dom server and the entire `ldap.server.dom` directory tree or one of its subtrees will be seen in the CommuniGate Pro Directory as some subtree.

Initially, the CommuniGate Pro server creates one Local Storage Unit `Main` that contains the entire directory. You may add additional storage units using the WebAdmin Interface.

The diagram below shows a directory stored in one Local Unit `Main`:
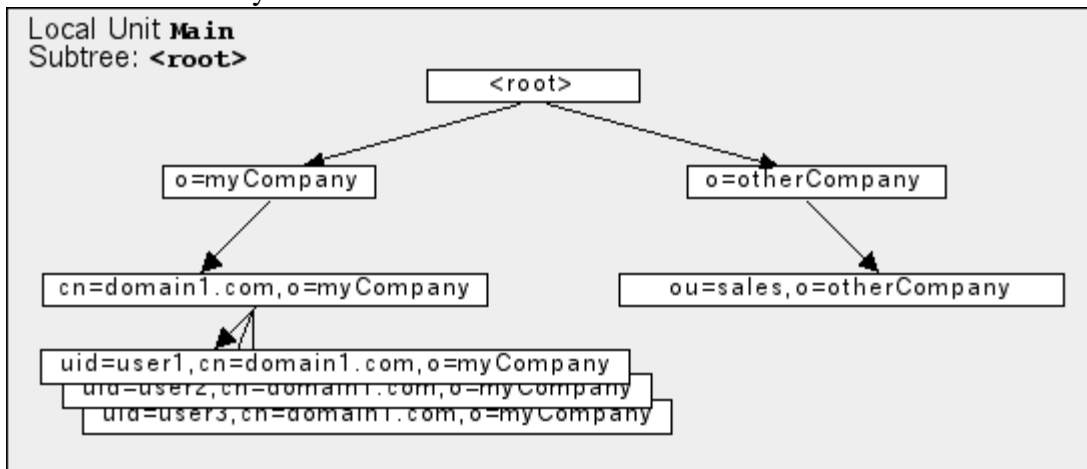


The diagram below shows the same directory stored in 2 Storage Units, with the entire `o=MyCompany` subtree stored in a separate Storage Unit `LDAP1`:
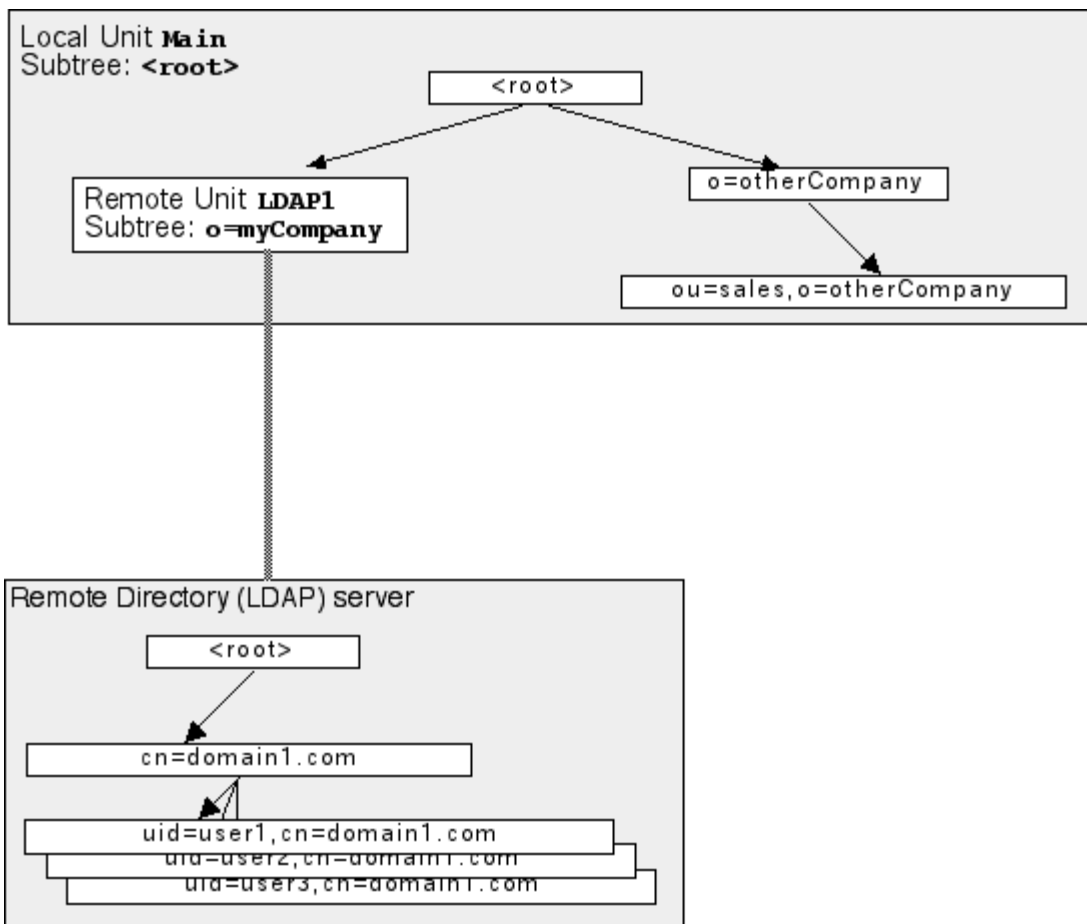
Example 1:

An external LDAP server `ldap1.com` has a subtree `o=MyCompany`, and you want to store all CommuniGate Pro Domain and Account records in that subtree. You can use the following settings:

- In the [Integration](#) settings, specify Base DN as `o=MyCompany,o=ldap1`
- Create a Remote Unit MYLDAP for the `o=ldap1` subtree.
- Enter `ldap1.com` as the server name, and an empty string as the Server Subtree in the MYLDAP Unit Settings.

Now, when the CommuniGate Pro Server tries to access a directory record for the Account `john` in the `domain1.com` Domain:

- The `uid=john,cn=domain1.com,o=MyCompany,o=ldap1` DN is formed.
- The Directory Manager detects that this record should reside on the MYLDAP Unit, and it asks that Unit to perform the requested operation on the record with the `uid=john,cn=domain1.com,o=MyCompany` DN (the Unit "mount point", `o=ldap1` is removed from the DN).
- The MYLDAP Unit sends the request for the `uid=john,cn=domain1.com,o=MyCompany` DN to the remote server `ldap1.com`.

Example 2:

An external LDAP server `ldap1.com` has a subtree `o=MyCompany`, and you want to store all CommuniGate Pro Domain and Account records in that subtree (the same situation as in the Example 1). You can use the following settings:

- In the [Integration](#) settings, specify Base DN as `o=ldap1`
- Create a Remote Unit MYLDAP for the `o=ldap1` subtree.
- Enter `ldap1.com` as the server name, and `o=MyCompany` as the Server Subtree in the MYLDAP Unit

Settings.

Now, when the CommuniGate Pro Server tries to access a directory record for the Account `john` in the `domain1.com` Domain:

- The `uid=john,cn=domain1.com,o=ldap1` DN is formed.
- The Directory Manager detects that this record should reside on the MYLDAP Unit, and it asks that Unit to perform the requested operation on the record with the `uid=john,cn=domain1.com` DN (the Unit "mount point", `o=ldap1` is removed from the DN).
- The MYLDAP Unit adds the Server Subtree suffix (`o=MyCompany`) to the DN, and then it sends the request for the `uid=john,cn=domain1.com,o=MyCompany` DN to the remote server `ldap1.com` .

Click the Directory link in the CommuniGate Pro Server WebAdmin Interface. The Directory Storage Units pages (realm) opens. You need to have the Directory access right to open these pages.

| Filter: | 2 of 2 selected |
|---------|------------------|
| **Unit Name** | **Subtree** |
| Main | <root> |
| node6 | o=node6 |

The <root> string is used to specify the name of the default Storage Unit (i.e. the Storage Unit that stores the root of the Directory Tree).

To create a new Storage Unit, type a name for the new unit (this name will be used for administrating only), the Distinguished Name (DN) for the subtree that should be stored in that unit, and click the Add Remote Unit or Add Local Unit button.

# Local Storage Units

Local Directory unit is a set of files containing unit data (records/entries), unit schema, unit settings, and unit modification journal.

Open the Directory WebAdmin page and click the name of a Local Storage Unit. The unit Settings page opens.

| Settings | | |
|----------|---|---|
| Subtree: | | |
| Log Level: | Major & Failures | |
| Enforce Schema | Search Results Limit: | 30 |
| Compose 'displayName' attribute | | |

`Log`
Use this setting to specify what kind of information the Local Storage Unit Manager should put in the Server Log.

`Enforce Schema`
When this option is selected, the Local Storage Unit Manager compares the structure of all new and updated records with the Unit Schema. If this option is disabled, then the Manager checks only the names of the record attributes and ensures that those attributes are included into the Unit Schema, but it does not enforce the objectClass-related restrictions.

```
Search Results Limit
```
This setting limits the maximum number of records a Directory Search operation can return.

```
Compose 'displayName' attribute
```
If this option is selected, the Local Storage Unit Manager checks if the `displayName` attribute is explicitly requested in a search operation, but this attribute is absent in a retrieved Directory record. In this case, the Manager composes the `displayName` attribute on-the-fly, using the `cn` attribute, or, if it is absent, the `uid` attribute.

Each Local Unit has its own Schema. See the [Schema](#) section for more details on Unit Schemas.

You can browse and modify the Local Unit Schema by retrieving and modifying the virtual record with the `cn=schema` DN. If the Local Unit is mounted on some Directory subtree, the DN for the Unit Schema record is `cn=schema,subtree`.

# Remote Storage Units

Click a Remote Storage Unit name on the Directory WebAdmin page to open the unit Settings page:

| Settings | |
|---|---|
| Log: | Major & Failures |
| LDAP Server Name: | |
| Security: | Disabled |
| Server Subtree: | |
| BIND DN: | |
| Bind Password: | |
| Search Filter: | |
| Protocol Version: | 2 |
| Channel Cache: | 10 |

```
Log
```
Use this setting to specify what kind of information the Remote Unit Manager should put in the Server Log.

```
LDAP Server Name
```
This field specifies the name or the IP address of the remote LDAP server that hosts the Storage Unit subtree. If the remote LDAP server uses non-standard TCP port, you can specify the Server Name as `servername:port`.

```
Security
```
If the TLSPort option is set, connections to the remote server will be established in the secure mode. The default port for secure connections is 636.
If the STARTTLS option is set, connections are established to the regular LDAP port, and then the STARTTLS LDAP command is sent to initiate secure (encrypted) communication.

```
Server Subtree
```
This field specifies the remote LDAP server subtree to be "mounted". When this setting is left empty, the entire remote directory becomes visible as a CommuniGate Pro Directory subtree.

```
BIND DN, BIND Password
```
These fields specify the Distinguished Name and Password to use for "binding" (logging into) the remote LDAP server. If these fields are left blank, the CommuniGate Pro will use anonymous access to the remote LDAP server.

```
Search Filter
```
This field contains an optional search filter (in the RFC 2254 format). This filter is included into all LDAP search operations sent to the LDAP server.

```
Protocol Version
```
Use this field to specify the LDAP protocol version supported with the remote LDAP server.

```
Channel Cache
```
Use this field to specify the number of "cached" TCP connections used to connect to the remote LDAP server.

---

# Remote Directory Root

In certain situations a CommuniGate Pro server should not keep any Directory data in its Local Storage units. Instead, all Directory records should be stored on a remote LDAP server (some other CommuniGate Pro server or a third-party Directory server). In this case, the CommuniGate Pro "root" should be stored in a Remote Storage Unit pointing to that external server. By default, the Directory "root" is stored in the `Main` Local Storage Unit.

To tell the CommuniGate Pro Server that the Directory "root" and the entire Directory tree is stored on a remote server, follow these steps:

- open the `Main` Storage Unit settings
- relocate the Unit to a fictitious subtree `o=dummy`
- create a Remote Storage Unit `RemoteRoot` specifying an empty string as its Subtree
- configure the `RemoteRoot` Storage Unit so it will access the proper remote LDAP server
- check that the remote directory is available (using the CommuniGate Pro Directory Browser)
- open the `Main` Storage Unit settings and click the Remove Unit button to remove this Storage Unit

---

# Binding to the Directory

Directory records can be (and usually are) protected from unauthorized access. When users want to access protected Directory data, they should authenticate themselves first. This process is called *binding* and successful authentication "binds" the user to a certain DN (distinguished name) in the Directory.

When a user tries to read or modify the Directory data, the binding DN is used to check the Directory Access Rights.

When a user accesses the Directory from a CommuniGate Pro WebUser Interface or a XIMSS session, the binding DN is the DN of the user Account record:
```
uid=accountname,cn=domainname,o=MyCompany.
```
See the Directory Integration chapter for the details.

When the Directory is accessed using the LDAP module, the client can authenticate itself using the CommuniGate Pro Account name and the Account password. In this case, the binding DN is the DN of the Account record.

Before converting the user account name into the account Directory record DN, the user account Server Access Rights are checked. If the account has the Directory access right, the special "master" bind DN is used instead of the user account record DN. Clients with the "master" bind DN have unlimited Directory access rights.

Any Directory DN can be used for LDAP binding. The directory record with the specified DN must exist, the record should contain the `userPassword` attribute, and the attribute value must match the supplied password string.

If a client has not authenticated itself, the special `anyone` bind DN is used.

# Access Right Records

The CommuniGate Pro Directory restricts client rights to read, search, and modify Directory records. The Directory contains a set of the Access Right records that allow and prohibit directory operations depending on the target directory subtree and on the client binding DN.

Open the Directory Access Rights page to set the Access Right records:

| Name | Target | Bind DN | Type | |
|------|--------|---------|------|------|
| | | | allow | **Edit** |
| | | | allow | **Edit** |
| | | | allow | **Edit** |
| | | | allow | |

Each Access Right record has:

- a name
- a target: the DN the record applies to; wildcard symbols ("*") can be used in Target strings
- a Bind DN: the client binding DN the record applies to; wildcard symbols ("*") can be used in Bind DN strings
- the record type: enabling or disabling
- a link to the Record specific access rights

The `Up` and `Down` buttons allow you to move the records in the table, increasing and decreasing record priorities.

When a client requests to perform a search, read, modify, or any other operation on a record or a subtree with a certain DN, the Access Right records are checked from top to the bottom. The server looks for an Access Right record that:

- has the Target field matching the DN specified in the client request
- has the Bind DN field matching the client binding DN
- has the operation (delete, create) matching the requested operation, or has the attribute matching the attribute used in the operation

When such an Access Right record is found, the record type specifies if the operation is allowed or prohibited. If no Access Right record is found, the operation is prohibited.

If the client binding DN is "master" (see above), all operations are allowed.

When a client requests a "read"-type operation, the procedure is repeated for all attributes the client wants to retrieve. If the operation is prohibited for all specified attributes, the read operation fails. Otherwise, the operation is performed, and the attributes the client has a right to retrieve are returned to the client.

If a client requests a "search"-type operation, the procedure is repeated for all attributes used in the search filter. If the search operation is prohibited for at least one of those attributes, the search operation fails. The RDNs of found records is checked. If the RDN attribute is not allowed to be read, the record is not returned to the client.

If a client requests a "rename"-type operation, the procedure is used twice: to learn if the client has a right to delete the original Directory record, then to learn if the client has a right to create a Directory record in the new location.

Special strings can be used in the Bind DN field:

`anyone`
> the Access Right record is applied for any BindDN, including the `anyone` (absent) BindDN.

`sibling`
> the Access Right record is applied if the Bind DN and the Target DN have the same parent DN. For example, the `uid=someuser,cn=domain1.com` and `uid=otheruser,cn=domain1.com` DNs are "siblings". This type of bind DN specifications is useful to grant CommuniGate Pro users access to the Directory records of other users in the same CommuniGate Pro Domain.

`parent`
> the Access Right record is applied if the Bind DN is a parent of the Target DN. For example, the `cn=domain1.com` DN is a parent of `uid=user1,cn=domain1.com` and `id=book1,uid=user1,cn=domain1.com` DNs.

`child`
> the Access Right record is applied if the Target DN is a parent of the Bind DN.

`self`
> the Access Right record is applied if the Target DN is the same as the Bind DN. This type of bind DN specification is useful to grant CommuniGate Pro Account users a right to modify their own directory record attributes.

To create an Access Right record, enter the record name, target DN, and bind DN into the last empty element of the Access Rights table and click the Update button. Use the `Up` buttons to set the record priority.

To remove an Access Right record, delete the record name and click the Update button.

The [Dynamic Cluster](#) environment supports the Cluster-wide Directory Access Rights. These Access Rights can be set on any Cluster Member and they are distributed to all Cluster Members.
The Cluster-wide Access Rights are applied after the Server-wide Access Rights.
To set the Cluster-wide Directory Access Rights, open the Access Rights page and click the Cluster-wide link.

---

# Access Right Specifications

To specify Directory Access Rights, open the Access Rights page and click the "specifications" link to open the Access Right record:

| Record-Level Rights | | | |
|---|---|---|---|
| prohibit | Record Deletion | prohibit | Record Creation |

These options specify if clients with the given Bind DN can create or delete records with the given Target DN.

| Attribute Reading |
|---|
| |
| prohibit: |

This field lists the data attributes that clients with the given Bind DN can read from the records with the given Target

DN. The attribute names should be comma-separated. To allow clients read all record attributes, use the asterisk (*) symbol.

| Attribute Searching |
|---|
| prohibit: |

This field lists the attributes that clients with the given Bind DN can use in filters when searching Target DN subtrees.

| Attribute Modifying |
|---|
| prohibit: |

This field lists the data attributes that clients with the given Bind DN can modify in the Target DN records.

Sample Access Right record:

Target DN
```
uid=*,cn=domain1.com
```
Bind DN
```
sibling
```
Type
```
allow
```
Readable Attributes
```
objectClass,officeEmail,roomNumber,cn,uid
```

This record allows all `domain1.com` users to read the `objectClass`, `cn`, `uid`, `officeEmail`, and `roomNumber` attributes from Directory records of other `domain1.com` Domain users.

Sample Access Right record:

Target DN
```
cn=domain1.com
```
Bind DN
```
child
```
Type
```
allow
```
Readable Attributes
```
objectClass,officeEmail,roomNumber
```
Searchable Attributes
```
cn,uid
```

This record allows all `domain1.com` users to search the Domain Directory subtree by `cn` (canonical name) and `uid`, but not by other readable attributes.

# Directory Browser

The CommuniGate Pro WebAdmin Interface includes a Directory Browser. Open the Directory realm and click the Browser link to open the Directory Browser page.

The Browser page includes the DN field:

| Distinguished Name |
|---|
| |

Use this field to type the DN of the Directory record/subtree you want to view and click the Go button. Click the Up button to remove the leftmost DN element and open the parent Directory record.

The next panel displays the Directory record with the specified DN:

| Attribute | Value |
|---|---|
| businesscategory | "Software development/technical support" |
| description | "\"Stalker Software, Inc. subsidiary\"" |
| o | "CommuniGate Systems" |
| objectclass | organization |
| seealso | "o=CommuniGate Systems, c=US" |
| telephonenumber | 676-555-1212 |

If the record with the specified DN could not be retrieved, this panel will contain the error message.

The next panel displays all record children.

| Subtree | |
|---|---|
| 300        Filter: | |
| **RDN** | **objectClass** |
| cn=aaa.dom | (top,organization,CommuniGateDomain) |
| cn=bbb.dom | (top,organization,CommuniGateDomain) |

Use the pop-up menu to limit the number of records displayed on the subtree panel.

To search for specific records, enter an LDAP filter string (in the RFC 2254 format) into the Filter field and click the Display button.

The table elements display children RDNs and object classes.

Click the child element RDN link to open the child record in the Directory Browser.

# Importing Directory Data

The CommuniGate Pro WebAdmin Interface allows the Server Administrator to import directory modifications from text files in the LDIF and "replog" formats:
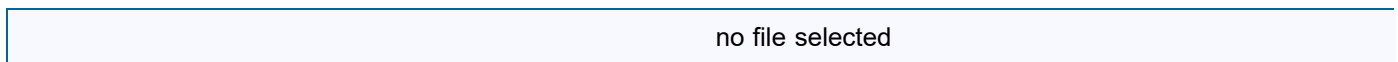
| no file selected |
|---|

To import data into the CommuniGate Pro Directory, click the Browse button and select an LDIF file on your workstation. Click the LDIF Import button to insert all records from the selected LDIF file.

To apply a set of record modifications to the CommuniGate Pro Directory, click the Browse button and select a "replog" file on your workstation. Click the LMOD Import button to apply all modifications from the selected file.

# Cluster-wide Directory Units

The [Dynamic Cluster](#) environment supports "Shared" or "Cluster-wide" Directory Units. These Units are "visible" on all Cluster members.

When Cluster-wide Unit settings are modified via any Cluster member, the new settings are automatically distributed to all other members.

When a Cluster-wide Remote Unit is used on any Cluster member, an outgoing LDAP request is generated directly from that member. All necessary synchronization tasks are performed on the remote LDAP server.

When a Cluster-wide Local Unit is used on the Cluster Controller, the request is performed locally. When a Cluster-wide Local Unit is used on a non-Controller Cluster member, the request is forwarded to the current Cluster Controller using the LDAP protocol (in the same way Remote Units relay requests to remote LDAP servers).

The Cluster-wide Unit data (settings and optional data files) are stored inside the [SharedDomains](#) file directory. When the current Cluster Controller stops or fails, and the Backup Controller assumes the Cluster Controller role, it re-mounts all Cluster-wide Local Units, and processes them as regular Local Units, while other Cluster members redirect requests to those Cluster-wide Local Units to this new Cluster Controller.

To create a Cluster-wide Unit, select the Cluster-wide checkbox (it appears in the Dynamic Cluster environment only).

---

| Directory | Schema | Integration |
|---|---|---|

# Directory Schema

- **Default Schema**
- **Record Attributes**
- **Object Classes**
- **Object Class Descriptor**

The CommuniGate Pro Local Storage Units support expandable Directory *Schema*.

Every Unit has its own Schema that specifies the data (object classes and attributes) that can be stored in that Directory Unit.

You can view and modify the Unit Schema using the Web Administration Interface. Open the Local Unit Settings page and click the Schema link. The Schema page will open and it will display all attributes and object classes defined for this Storage Unit.

## Default Schema

When a new Local Storage Unit is created, the Default CommuniGate Pro Schema is automatically created for that Local Storage Unit.

## Record Attributes

The first part of the Schema page is the list of all record Attributes that can be used in this Storage Unit.

| Attributes | | |
|---|---|---|
| Name: | | |
| Object ID: | | |

| Name | Object ID | Syntax |
|---|---|---|
| objectClass | 2.5.4.0 | |
| aliasedObjectName | 2.5.4.1 | |
| cn | 2.5.4.3 | |

| | |
|---|---|
| sn | 2.5.4.4 |
| c | 2.5.4.6 |
| l | 2.5.4.7 |
| st | 2.5.4.8 |
| ............................................ | |
| serverAccessRights | 2.5.4.10103 |
| webUserSettings | 2.5.4.10110 |

This table lists all attributes defined in the Local Unit Schema.

You can add new attributes to the Unit Schema. Type the new attribute name and (optionally) new attribute Object ID (OID) into the text fields and click the Add Attribute button.

---

# Object Classes

The second Schema page table lists all object classes defined in this Local Unit Schema:

| Object Classes | | | | |
|---|---|---|---|---|
| Name: | | | | |
| Object ID: | | | | |
| Parent: | top | | | |
| **Name** | **Object ID** | **Parent** | **Required Attributes** | **Optional Attributes** |
| top | 2.5.6.0 | | objectClass | |
| alias | 2.5.6.1 | top | aliasedObjectName | |
| country | 2.5.6.2 | top | c | |
| organization | 2.5.6.4 | top | o | street, postOfficeBox, telephoneNumber, facsimileTelephoneNumber, userPassword, dc |
| organizationalUnit | 2.5.6.5 | top | ou | street, postOfficeBox, telephoneNumber, facsimileTelephoneNumber, userPassword, dc |
| person | 2.5.6.6 | top | cn, sn | description, telephoneNumber, facsimileTelephoneNumber, userPassword |
| organizationalPerson | 2.5.6.7 | person | | |
| inetOrgPerson | 2.16.840.1.113730.3.2.2 | organizationalPerson | | uid, mail |
| CommuniGateDomain | 2.5.1000.0 | organization | | accessModes, autoSignup, RPOPLimit, accountsLimit, storageLimit, listsLimit, trailerText, webBanner, mailRerouteAddress, foldering, mailToAllAction, mailToUnknown, centralDirectory, accountsLogLevel, |

| | | | | mailboxesLogLevel, domainAccessModes, IPMode, IPAddresses, webUserCache, externalLocation, externalLockType, osUserName |
|---|---|---|---|---|
| CommuniGateAccount | 2.5.1000.1 | inetOrgPerson | I | maxAccountSize, externalINBOX, hostServer, maxWebSize, maxWebFiles, accessModes, rulesAllowed, RPOPAllowed, PWDAllowed, mailToAll, addMailTrailer, addWebBanner, passwordEncryption, defaultMailboxType, useAppPassword, useSysPassword, useExtPassword, requireAPOP, recoverPassword, storageLocation |
| CommuniGateAccountTemplate | 2.5.1000.2 | CommuniGateAccount | | initialMailboxes, initialSubscription |
| CommuniGateAlert | 2.5.1000.20 | top | alertTimeStamp, alertText | |
| CommuniGateAccess | 2.5.1000.21 | top | serverAccessRights | |
| CommuniGateWebUser | 2.5.1000.22 | top | webUserSettings | |

To add an objectClass to the Local Unit Schema, enter the new class name, (optionally) class object ID (OID), and select the parent objectClass from the pop-up menu listing all existing classes. Click the Add Class button to add a new class to the Schema.

Click the class name link to open the Class [Descriptor page](#).

---

# Object Class Descriptor

You can click the class name on the Local Unit Schema page to open the Object Class Descriptor page:

| objectClass | |
|---|---|
| **Required Attributes** | **Optional Attributes** |
| cn, sn | description, telephoneNumber, facsimileTelephoneNumber, userPassword |
| **Parent: parentClass** | |
| objectClass | |

This table lists the Class attributes - required and optional.

The first part of the table lists the attributes defined for the class itself, while the second part of the table lists the attributes defined in the class parent classes.

You can extend your Schema by adding more attributes to a Schema Class. Select the attribute name from the pop-up menu and click either the Add Required Attribute or Add Optional Attribute button.

| Directory | Schema | **Integration** |

# Directory Integration

- **Directory Integration Concept**
- **Attribute Renaming**
- **Domains Subtree**
- **Custom and Public Info Account Settings**
- **Integrating Regular Domains**
- **LDAP-based Provisioning**
- **Directory-Based Domains**
- **Shared (Multi-Server) Directory**
- **Distributed Domains (Directory Routing)**
- **Directory Integration in a Cluster**

CommuniGate Pro Directory can be used to store any information. One of the Server features is integration of the CommuniGate Pro Directory and CommuniGate Pro Domains. The integration level is selected on a per-Domain basis.

A Server Administrator can control how CommuniGate Pro Domains are integrated with the Directory. Open the Domains page and follow the Directory Integration link.

## Directory Integration Concept

CommuniGate Pro Domains can use the following levels of Directory integration:

- No integration; the Domain and the Domain Accounts settings are stored in `.settings` files, and when an Account is created, updated, renamed, or removed, Directory is not updated.
- Synchronized; the Domain and Domain Account settings are stored in the `.settings` files; each Account has a Directory record that stores *some* of the Account settings as attributes:
  - the Account name in the `uid` attribute
  - the Account Real Name in the `cn` attribute
  - the Account Certificate in the `userCertificate` attribute (if exists)
  - the hosting Server Main Domain name in the `hostServer` attribute (this attribute is needed to implement Directory-based Static Clusters)
  - an optional set of custom (and "public info") settings/attributes.

  When an Account is created, renamed, removed, or updated, the Directory is automatically updated.
- Directory-based. The Domain and the Domain Account settings are stored in the Directory records. The `.settings` files are not created, and the Server retrieves all settings information from the Directory.

Finally, the CommuniGate Pro can use regular (non Directory-Based) domains, but still allow Account provisioning via LDAP, so it looks like the Directory-Based Domains are used and LDAP commands can create and update Account. This feature is called LDAP-based Provisioning.

# Attribute Renaming

Some Domain and Account settings names may not match the standard attribute names used in the Directory Schema. For example, the Account setting `Real Name` has to be stored in the Directory as the `cn` (common name) attribute, and the custom settings `surname` and `city` (see below) should be stored as attributes `sn` and `l`.

When you need to add an attribute to your Directory Schema, always try to use attribute names specified in one of the LDAP Internet Standards (RFCs). If this attribute should be used for Directory Integration (i.e. it will be used to store some Domain or Account setting value), you may want to use the Attribute Renaming capability to "map" CommuniGate Pro Domain or Account setting name on some Directory Attribute name.

Use the Attributes Translation table to specify the name translation rules:

| Attributes Translation | |
|---|---|
| **Name in CommuniGate** | **Name in Directory** |
| | |

**Note:** The Attributes Translation feature works only for the Directory Integration component of the CommuniGate Pro Server. If you access the CommuniGate Pro Directory directly (via the LDAP module, for example), no translation takes place: LDAP clients should specify the Directory Attribute names, and the returned records have Directory Attribute names, not CommuniGate Pro Domain and Account setting names - "cn", not "RealName" and "userPassword", not "Password".

# Domains Subtree

For each regular CommuniGate Pro Domain with the Directory setting set to `Keep in Sync`, and for each Directory-Based Domain a Directory Subtree is created. This Subtree has the Domain record as its root, and all Domain Account records as the Subtree elements ("leaves"). For Directory-based Domains, additional elements are created to store Account aliases, Domains settings, etc.

The Domain Subtree panel allows you to specify the location of Subtrees created for each CommuniGate Pro Domain:

| Domain Subtree | |
|---|---|
| Base DN: | |
| Domain RDN Attribute: | |
| Domain objectClass: | |
| UID Subtree: | |

`Base DN`
> This field specifies the "base" DN for all Domains in the Central Directory. You may want to set it as:

```
        o=your company name
```

so each CommuniGate Pro Domain will have the following DN:

```
        cn=domain name,o=your company name
```

When a Domain is placed into the Directory, a record with its DN is created. If the Base DN does not exist, the Directory Manager may return an error. Use the Create It button to create an empty record with the Base DN.

If you are an ISP you may want to give each Domain you host the top-level DN:

```
        cn=domain name
```

In this case, specify an empty string in the Base DN field.

Domain RDN attribute

This field specifies the attribute name to use for Domain record RDNs. In most cases, the default value (cn) is the best choice. However, you can change that to the name of any other attribute defined in the Directory schema. If you set this name to o, the CommuniGate Pro Domain records will have the following DNs:

```
        o=domain name,base DN
```

**Note:** If you specify the string dc as the Domain RDN attribute, then the DN for a CommuniGate Pro Domain mail.domain.dom will be composed as dc=mail,dc=domain,dc=dom.

Domain objectClass

This field specifies the *objectClass* for CommuniGate Pro Domain records in the Directory. The CommuniGateDomain objectClass defined in the CommuniGate Pro Directory Manager schema is the default value. If you choose to select a different objectClass, make sure it exists in your Directory schema.

For regular Domain, the Domain Directory record is empty. As a result, you may use any objectClass that can store the cn attribute (or the attribute you have specified in the Domain RDN attribute setting).

For Directory-based Domains, the Domain Directory record contains all Domain settings, so the objectClass for these records should support all attributes included into the CommuniGateDomain objectClass.

UID subtree

If this field is empty, then the Domain Object (Account, Group, List, Forwarder) records are stored in the directory using the following DNs: uid=*objectName*,*domain DN*.
If the base DN is o=mycompany, and the Domain RDN attribute is cn, then the directory record for the user1 Account in the domain1.dom Domain will have the following DN:
uid=user1,cn=domain1.dom,o=mycompany
The UID subtree parameter allows you to place the objects "below" the domain tree. If the UID subtree is set to ou=People, then the record for the same Account will have the following DN:
uid=user1,ou=People,cn=domain1.dom,o=mycompany
If the Domain RDN attribute is set to dc, then the record for the same Account will have the following DN:
uid=user1,ou=People,dc=domain1,dc=dom,o=mycompany

Example:

BaseDN is an empty string, Domain RDN Attribute is cn, and three CommuniGate Pro domains (domain1.dom, domain2.dom, and domain3.dom) have been created:
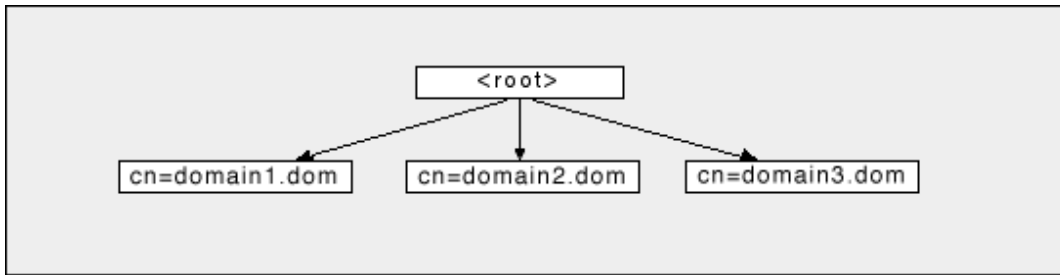
To search for Accounts in these Domain subtrees, LDAP clients should have the string

```
cn=domainN.dom
```

specified in their "Base Object" or "Search Base" settings.

Example:

`BaseDN` is `o=acme`, `Domain RDN Attribute` is `cn`, and three CommuniGate Pro domains (domain1.dom, domain2.dom, and domain3.dom) have been created:
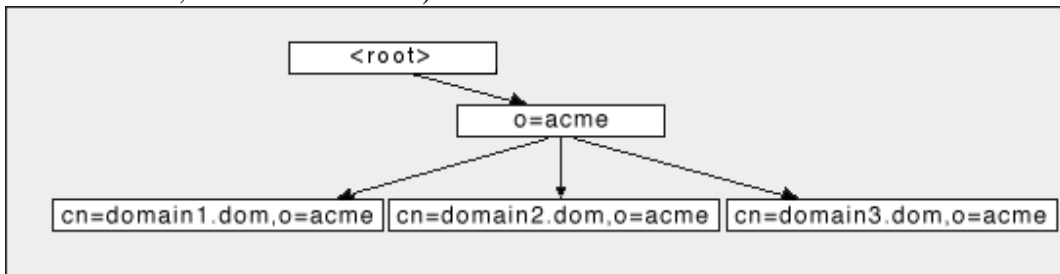


To search for Accounts in these Domain subtrees, LDAP clients should have the string

```
cn=domainN.dom,o=acme
```

specified in their "Base Object" or "Search Base" settings.

Example:

`BaseDN` is `o=acme`, `Domain RDN Attribute` is `dc`, and three CommuniGate Pro domains (domain1.dom, domain2.dom, and domain3.dom) have been created:
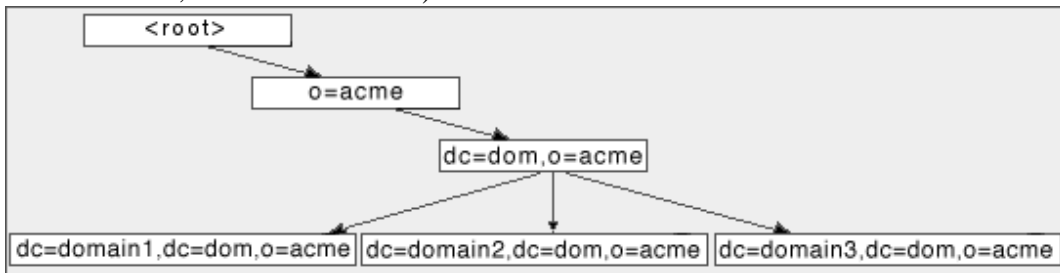


To search for Accounts in these Domain subtrees, LDAP clients should have the string

```
dc=domainN,dc=dom,o=acme
```

specified in their "Base Object" or "Search Base" settings.

After you have decided how to organize your Domains Subtree, you can create additional Directory [Storage Units](#) to store your Domain and Account data in several units (if necessary). For example, if you want to use your CommuniGate Pro Directory Manager to store information not related to CommuniGate Pro Accounts, and you want all Domain and Account information to be stored either on a remote LDAP server or in a dedicated Local Storage Unit, you can create a Storage Unit `MyDomains` for the Directory Integration Base DN subtree (`o=acme` in the examples listed above). In this case, all Domains and Account records will be stored in that `MyDomains` Storage Unit (in a separate local unit or on a remote LDAP server), while all records that do not have the `o=acme` suffix will be stored in other Storage Units:
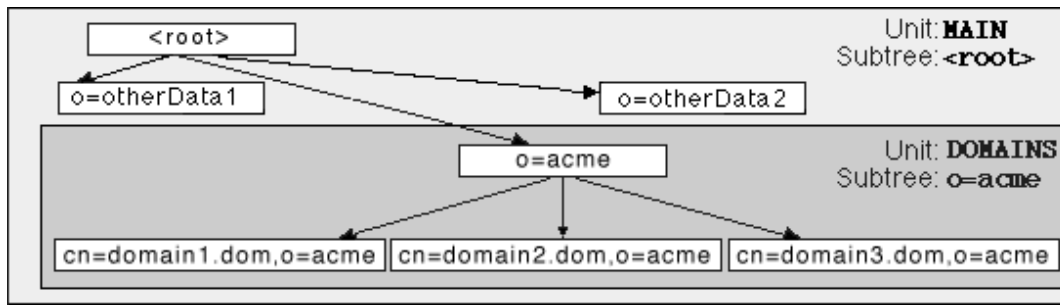
**Note:** If you change any Domain Subtree setting, the existing Subtree is not modified. Carefully select the proper values for the Domain Subtree settings before you start any Directory Integration activity. If you need to change these settings later, it is your responsibility to move the existing Domain Subtree to the new location (specified with the new BaseDN) and/or to change RDNs of the existing Domain records (if you have changed the Domain RDN Attribute setting).

---

# Custom Account Settings

The CommuniGate Pro Server has a predefined set of Account Settings (see the Accounts section for more details). The Directory Integration settings include a panel that allows you to specify additional, Custom settings for CommuniGate Pro Accounts:

| Custom Account Settings | |
|---|---|
| **System** | **Public Info** |
| | |
| | |
| | |

You can use these Custom Account Settings to store additional information about your users: locations, phone numbers, demographic data, etc.

The System custom settings can be modified by Server administrators and Domain administrators with the Basic Domain Access Right.

The Public Info custom settings can be modified by the users themselves.

To add a Custom Setting, type its name into the last (empty) field and click the Update button.

Additional (custom) Account Settings are stored in Account Directory records (these records have the CommuniGateAccount objectClass).

When you select a name for a new Custom Account Setting, either use a name of an attribute already specified for CommuniGateAccount object class in the Directory Schema, or use the Directory Integration Attribute Renaming feature and map the new Custom Account Setting name onto a name of any already specified attribute.

Example:

> To add the `telephoneNumber` setting to all CommuniGate Pro Accounts, add the name `telephoneNumber` to the Custom Account Settings table.
> If a Local Storage Unit is used to store CommuniGate Pro Domains and Accounts subtree, no additional action is needed: the `telephoneNumber` attribute is already included into the CommuniGateAccount object class description in all Local Unit Schemas.

Example:

> To add the `surname` setting to all CommuniGate Pro Accounts, add the name `surname` to the Custom Account Settings table, and add the pair (`surname`, `sn`) to the Attribute Renaming table, so the `surname` Account Settings will be stored in Directory records as `sn` attributes.
>
> If a Local Storage Unit is used to store CommuniGate Pro Domains and Accounts subtree, no additional action is needed: the `sn` attribute is already included into the CommuniGateAccount object class description in all Local Unit Schemas.

Example:

> To add the `BirthDay` setting to all CommuniGate Pro Accounts, add the name `BirthDay` to the Custom Account Settings table.
>
> If a Local Storage Unit is used to store CommuniGate Pro Domains and Accounts subtree, add the `BirthDay` attribute to the Local Unit Schema, and add the newly created `BirthDay` attribute name to the list of Optional Attributes of the CommuniGateAccount object class.
>
> If a Remote Storage Unit is used to store CommuniGate Pro Domains and Accounts subtree, update the Directory Schema on the remote LDAP server to allow directory records of the CommuniGateAccount object class to include the `BirthDay` attribute.

**Note:** Account records in the Directory always contain the `sn` attribute to make them compatible with the standard LDAP Directory Schema. If you do not include this attribute into the Custom Account Settings set, CommuniGate Pro stores account records with the `sn` attribute containing the `none` string.

After you have specified some Custom Account Settings, their names appear on the Account Settings pages. You can use those pages or [CLI](#) to add and update the Custom Setting values for all CommuniGate Pro Accounts:

| | |
|---|---|
| Real Name: | |
| surname: | |
| city: | |
| CommuniGate Password: | |
| telephoneNumber: | |

**Note:** if you rename a custom attribute name or remove it, the attribute values are not modified in the Directory - you are effectively changing the Directory Integration parameters, not the Directory data itself. To update the actual Directory data (for example, to remove all `telephoneNumber` attribute values from the Directory), use LDAP utilities and/or applications.

---

# Integrating Regular Domains

Regular CommuniGate Pro Domains do not rely on Directory data. All Domain and Account settings are stored in files inside the CommuniGate Pro file directories and CommuniGate Pro Server reads those files when it needs to retrieve Domain and Account settings. Regular Domains can store copies of **some** Account Settings in Directory records.

The directory record for a Regular Domain is created when the Server needs to store a directory record for any object in that Domain. For example, when the Server needs to create a directory record for the Account `john` in the `dom1.dom` Domain, it creates the `cn=dom1.dom` record first (if it does not exist), and then the Server creates the `uid=john,cn=dom1.dom` record for the Account `john`.

When the [Directory Integration](#) Domain Setting is set to `Keep In Sync`:

- A directory record is created for each object (Account, Group, Mailing List, Forwarder) created in that Domain.
- A directory record is removed when an object is removed from that Domain.
- Directory record DNs are renamed when Domain objects are renamed.
- Directory records are updated when Domain Accounts settings are updated.

None of these actions takes place when the Domain Directory Integration settings is set to Disabled.

The following diagram illustrates what happens when the `dom1.dom` Domain has the Directory Integration option set to `Keep In Sync`, and an Account is created in that Domain:
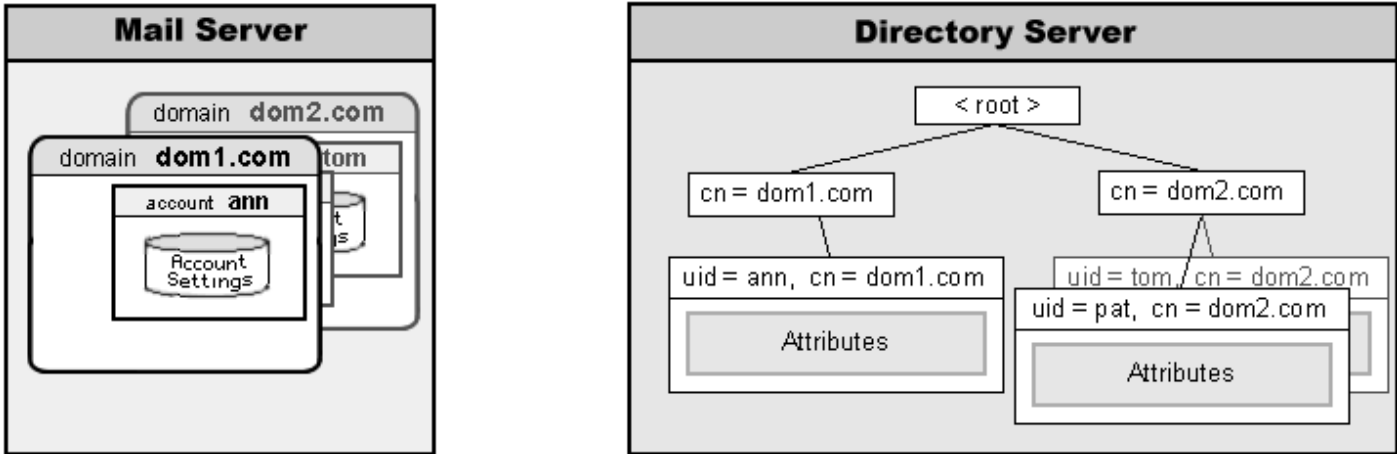


In this example:

- The WebAdmin or CLI interface is used to create the `john` Account in the `dom1.dom` Domain.
- The Account `john` is created, and the supplied settings (together with the Account Template) are used to compose the initial Account Settings.
- The Account Manager executes the AddRecord Directory operation to create a record in the Directory.
  The Directory record DN is composed using the global Directory Integration Settings. If the default settings are used, the Directory record DN is `uid=john,cn=dom1.dom`.
  Some of the initial Account Settings are converted into Directory attributes and stored into the newly created Directory record.

Now Directory search operations (initiated with an LDAP client or the WebUser Interface) can display the record for the newly created account.

Directory records for Regular Domain Accounts contain the following attributes:

- `uid` - the Account name
- `cn` - the Account "Real Name"
- `sn` - an empty string if this attribute is not included into Custom Account Settings
- `hostServer` - the Main Domain name of the CommuniGate Pro Server that hosts this Account
- `userCertificate` - the Account Certificate (if exists)
- custom settings (see above)
- `userPassword` - the Account password (*optionally*)
- other standard settings - (*optionally*)

The Regular Domains panel located on the Directory Integration page of the WebAdmin Interface allows you to specify these options:

| Regular Domains | |
|---|---|
| | Passwords |
| Copy into Account records: | Standard Settings |

`Copy Password`
> If this option is selected, the Directory records for Regular Domain Accounts will contain the Account Password attribute (usually it is renamed into the `userPassword` attribute).

`Copy Standard Settings`
> If this option is selected, the Directory records for Regular Domain Accounts will contain all CommuniGate Pro standard Settings for those Accounts (excluding the RealName and userCertificate settings that are always being stored and the Password setting that it controlled using a separate option).

The following diagram illustrates what happens when the `dom1.dom` Domain has the Directory Integration option set to `Keep In Sync`, and Domain Account settings are updated:
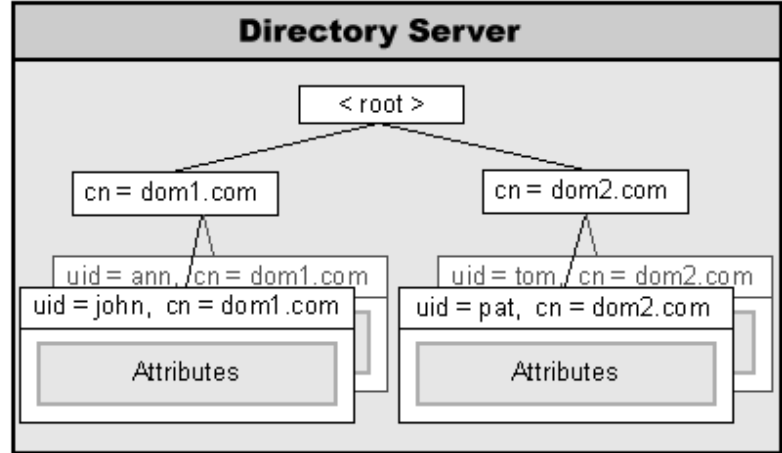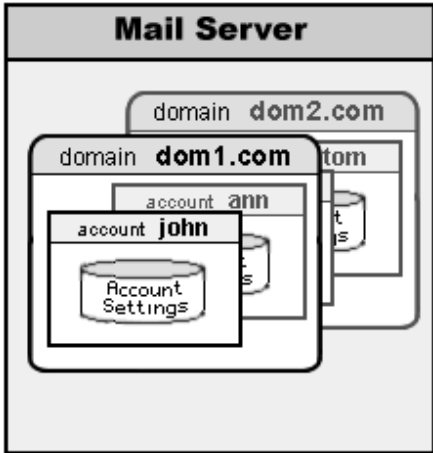


In this example:

- The UpdateAccount operation is initiated with a WebAdmin or CLI Interface command.
- The Account `john` Settings are modified using the supplied new settings and the updated settings are stored in the CommuniGate Pro Account data files.
- If the `dom1.dom` Domain Directory Integration setting is set to Keep In Sync, the Account Manager executes the UpdateRecord Directory operation to update the Account record in the Directory.

**Note:** It is important to understand that Directory Integration for Regular Domains is a one-way relationship: if you change attributes of Account records in the Directory (using any LDAP utility), the actual Account Settings will not be modified - CommuniGate Pro always uses data in the settings files, and never reads data from the Directory when it needs to retrieve settings for Regular Domains or settings for Accounts in those Domains. The CommuniGate Pro Manager for regular Domains and Accounts only updates the Directory, but it never reads the Account record data back from the Directory.

The following diagram illustrates what happens when the `dom1.dom` Domain has the Directory Integration option set to `Keep In Sync`, and a Domain Account is renamed:
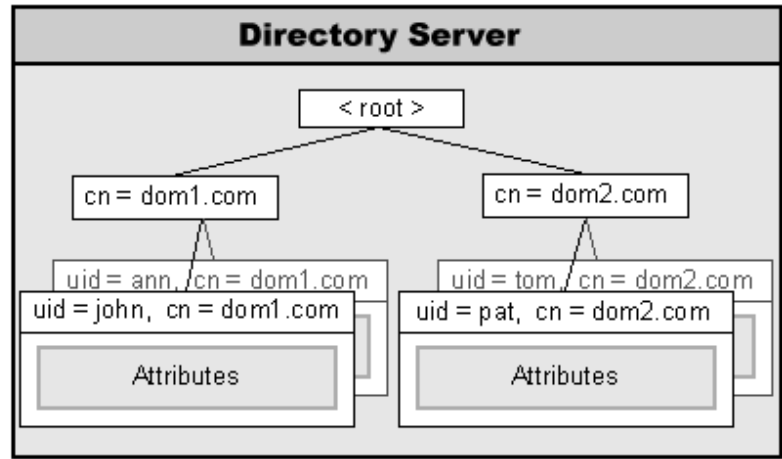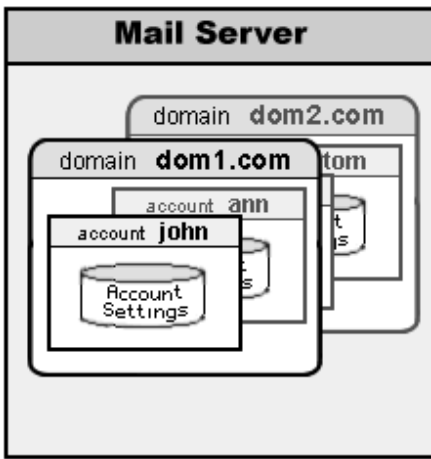
In this example:

- The RenameAccount operation is initiated with a WebAdmin or CLI Interface command.
- The Account john and its files are renamed.
- If the dom1.dom Domain Directory Integration setting is set to Keep In Sync, the Account Manager executes the RenameRecord (modifyDN) Directory operation to rename the Account record in the Directory.

The following diagram illustrates what happens when the dom1.dom Domain has the Directory Integration option set to Keep In Sync, and a Domain Account is removed:
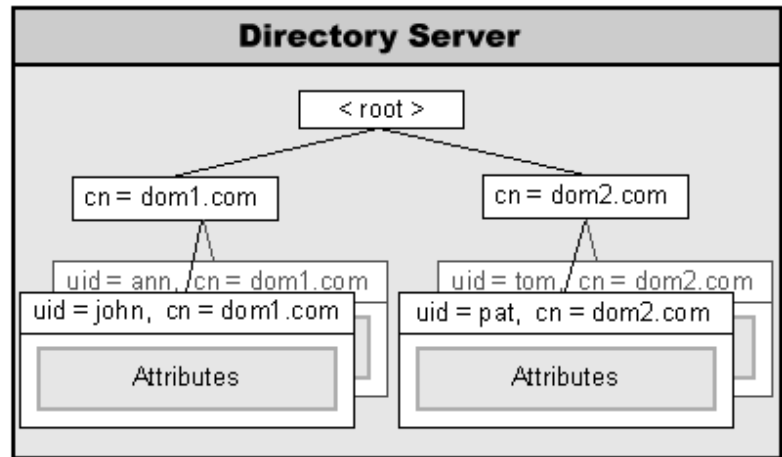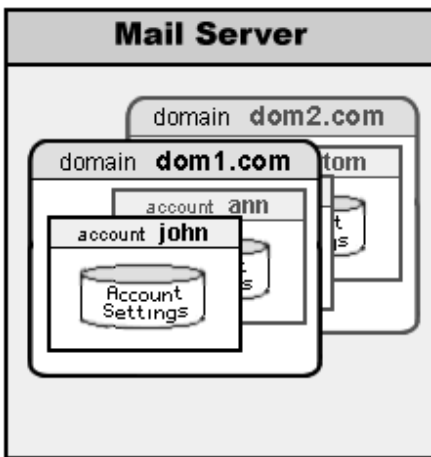


In this example:

- The RemoveAccount operation is initiated with a WebAdmin or CLI Interface command.
- The Account john and its files are removed.
- If the dom1.dom Domain Directory Integration setting is set to Keep In Sync, the Account Manager executes the DeleteRecord Directory operation to remove the Account record from the Directory.

The Directory Integration panel on the Domain Settings page has the Delete All button. Use this button to delete the Domain record and all Domain Object records from the Directory. The operation deletes only those records that contain the hostServer attribute and that attribute value is the same as the Main Domain name of this CommuniGate Pro Server.

The Directory Integration panel on the Domain Settings page has the Insert All button. Use this button to create a directory record for this Domain and to create directory records for all Domain Objects.

**Note:** If you have created several Accounts in the regular Domain when its Directory Integration setting was set to Disabled, the Directory does not contain records for those Accounts. If later you switch that setting to Keep In Sync, you will see error reports when you try to rename, remove, or update those Accounts: the Server tries to update the Directory records for those Accounts, but the Directory records do not exist.

Before you switch the Directory Integration setting from Disabled to Keep In Sync, click the Delete All button, and

then click `Insert All` button to synchronize the Directory and the current Domain Objects set.

# LDAP-based Provisioning

CommuniGate Pro allows you to use the LDAP protocol to create, update, rename, and remove Accounts:

| LDAP direct Provisioning |
| --- |
| Disabled |
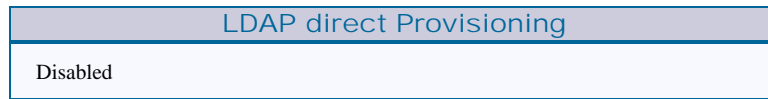
When this option is enabled, the LDAP module checks the names (DNs) specified in update operations. If the DN looks like a DN of a CommuniGate Pro Account, the LDAP module does not perform the requested operation with the Directory. Instead, it executes the CreateAccount, UpdateAccount, RenameAccount, or RemoveAccount operations for the specified Account and Domain.

The diagram below illustrates how the LDAP AddRecord operation works in this case:
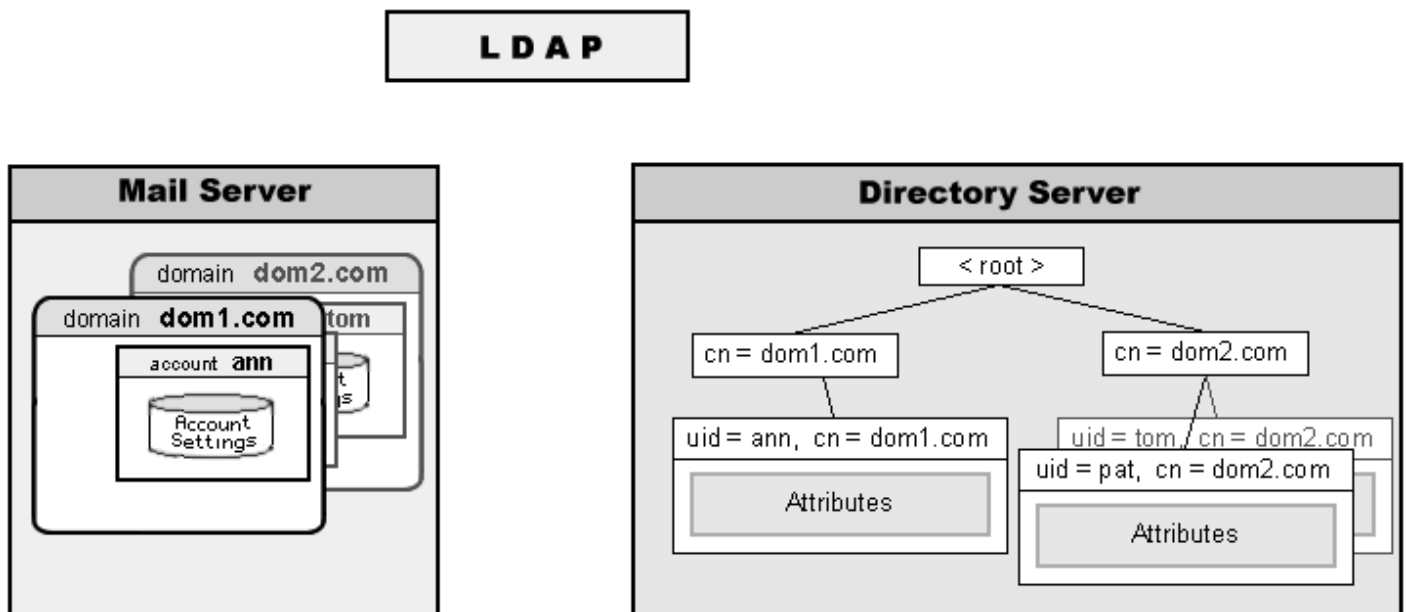


In this example:

- The LDAP module received an AddRecord request from an LDAP client. The client asks the LDAP module to create a record with the `uid=john,cn=dom1.dom` DN.
- The LDAP module checks the DN and sees that it looks like the record DN for the CommuniGate Pro Account `john` in the CommuniGate Pro Domain `dom1.dom`, and the CommuniGate Pro Domain `dom1.dom` does exist.
- Instead of performing the requested AddRecord operation with the Directory, the LDAP module executes the `CreateAccount(john)` operation in the `dom1.dom` Domain.
- The Account `john` is created, and the supplied LDAP attributes (together with the Account Template) are used to compose the initial Account Settings.
- If the `dom1.dom` Domain Directory Integration setting is set to Keep In Sync, the Account Manager executes the AddRecord Directory operation to create a record in the Directory.

**Note:** the Directory Integration settings are used to convert LDAP record attribute names into the CommuniGate Pro attribute names. For example, the LDAP AddRecord request can contain the `cn` attribute. This attribute is stored in the Account settings as the Account RealName setting. When the Account Manager adds a record to the Directory, it converts the RealName Account setting back into the `cn` record attribute.

**Note:** all LDAP AddRecord request attributes will be stored as the Account Settings if the LDAP client has authenticated itself as an Account with All Domain and Account Settings access right. But only the attributes specified with the

Directory Integration parameters will be copied into the new Directory record. The Directory record will also contain the attributes not included into the original LDAP AddRecord request, but specified in the Account Template.

**Note:** the LDAP Provisioning feature detects the `unixPassword` attributes and converts them into `Password` settings after adding a leading `0x02` byte. See the [Account Import](#) section for the details.

The following diagram illustrates how the LDAP ModifyRecord operation can be used to modify Account Settings:
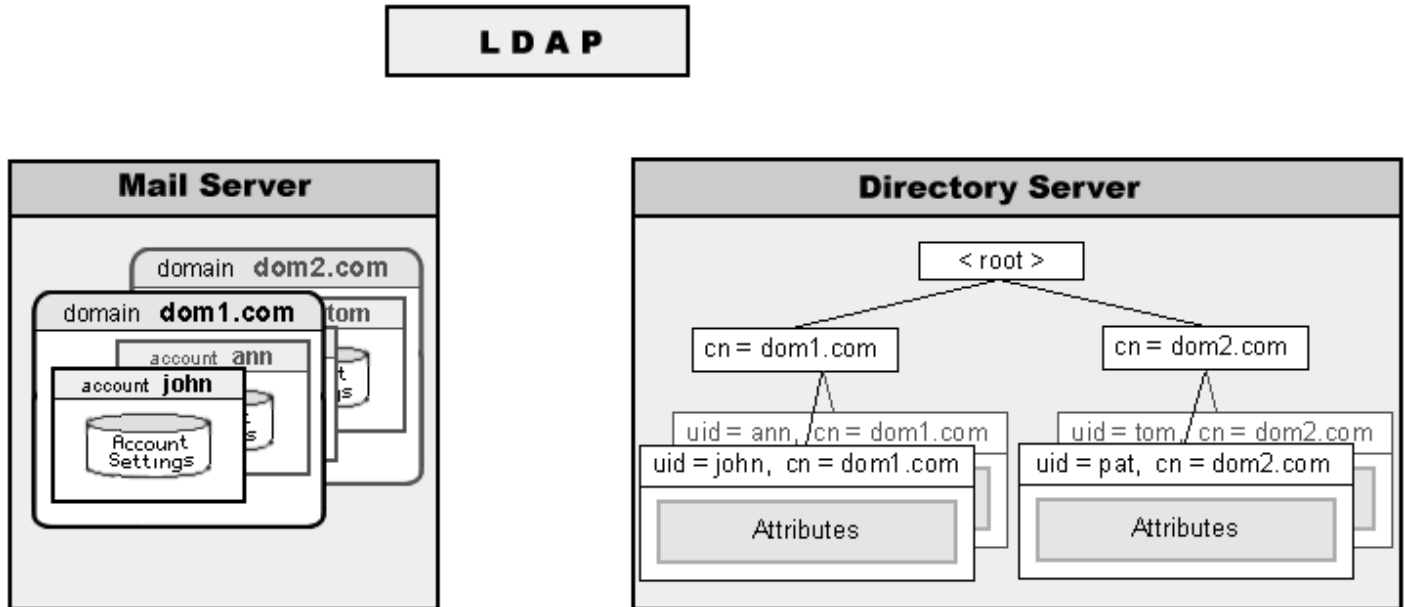


In this example:

- The LDAP module received a ModifyRecord request from an LDAP client. The client asks the LDAP module to update a record with the `uid=john,cn=dom1.dom` DN.
- The LDAP module checks the DN and sees that it looks like the record DN for the CommuniGate Pro Account `john` in the CommuniGate Pro Domain `dom1.dom`, and the CommuniGate Pro Domain `dom1.dom` does exist.
- Instead of performing the requested ModifyRecord operation with the Directory, the LDAP module executes the `UpdateAccount(john)` operation in the `dom1.dom` Domain.
- The Account `john` Settings are modified using the supplied LDAP attributes.
- If the `dom1.dom` Domain Directory Integration setting is set to Keep In Sync, the Account Manager executes the ModifyRecord Directory operation to modify a record in the Directory.

The following diagram illustrates how the LDAP ModifyDN operation can be used to rename Accounts:

In this example:

- The LDAP module received a ModifyDN request from an LDAP client. The client asks the LDAP module to change the `uid=john,cn=dom1.dom` record DN.
- The LDAP module checks the DN and sees that it looks like the record DN for the CommuniGate Pro Account `john` in the CommuniGate Pro Domain `dom1.dom`, and the CommuniGate Pro Domain `dom1.dom` does exist. It also checks that the new name (`uid=john1`) looks like some CommuniGate Pro Account record DN.
- Instead of performing the requested ModifyDN operation with the Directory, the LDAP module executes the `RenameAccount(john,john1)` operation in the `dom1.dom` Domain.
- The Account `john` is renamed into `john1`.
- If the `dom1.dom` Domain Directory Integration setting is set to Keep In Sync, the Account Manager executes the ModifyDN Directory operation to change the Account record DN in the Directory.

The following diagram illustrates how the LDAP DeleteRecord operation can be used to remove Accounts:
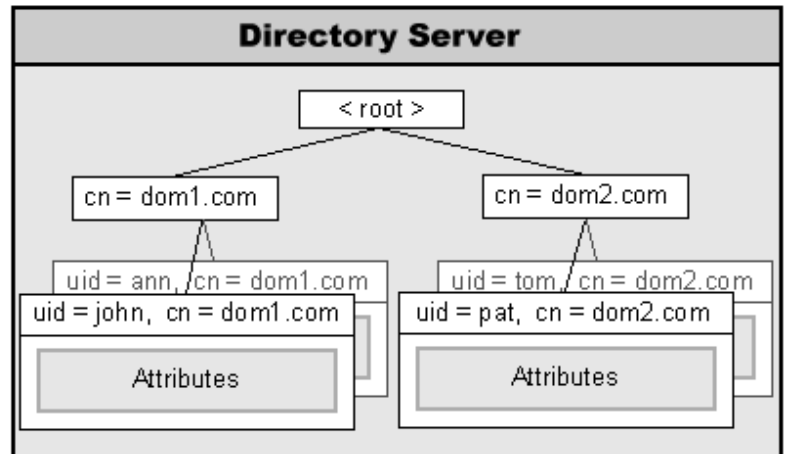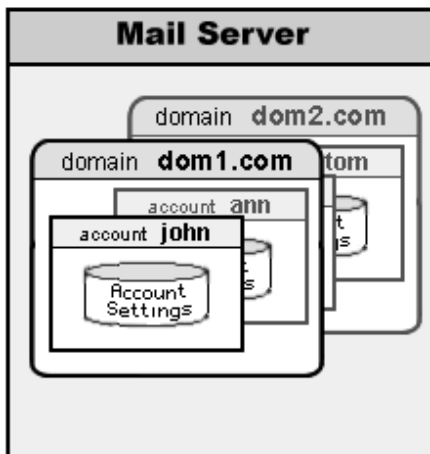


In this example:

- The LDAP module received a DeleteRecord request from an LDAP client. The client asks the LDAP module to delete the `uid=john,cn=dom1.dom` record.
- The LDAP module checks the DN and sees that it looks like the record DN for the CommuniGate Pro Account `john` in the CommuniGate Pro Domain `dom1.dom`, and the CommuniGate Pro Domain `dom1.dom` does exist.

- Instead of performing the requested DeleteRecord operation with the Directory, the LDAP module executes the `DeleteAccount(john)` operation in the `dom1.dom` Domain.
- The Account `john` is deleted.
- If the `dom1.dom` Domain Directory Integration setting is set to Keep In Sync, the Account Manager executes the DeleteRecord Directory operation to remove the Account record DN from the Directory.

When LDAP Provisioning is enabled, the LDAP module uses special processing for some search requests, too. If

- the search request Base DN looks like a DN of some CommuniGate Pro Account, and
- the search request "scope" parameter is "base",

the LDAP module calls the Account Manager directly and retrieves the effective settings for the specified Account (i.e. the Account settings as well as all its Default Settings). The module converts these settings into Directory attributes, and sends them back to the LDAP client as a search result record.
If the authenticated LDAP user does not have the Domain Administrator access right for the target Account Domain, only the Real Name, User Certificate, Custom, Public Info, and mail attributes are retrieved.

# Directory-Based Domains

The CommuniGate Pro Server software implements Directory-based Domains. Directory-based Domains and all their Accounts keep all their settings in the Directory - there is no `.settings` files for those Domains and Accounts.

For each Directory-based Domain a Directory record of the CommuniGateDirectoryDomain objectClass is created. This record stores all [Domain Settings](link).
DNs for Directory-based Domains are built in the same way they are built for Regular Domain records.

For each account in a Directory-based Domain a Directory record of the CommuniGateAccount objectClass is created. This record stores all [Account Settings](link) (including the Custom Settings).
DNs for accounts in the Directory-based Domains are built in the same way they are built for Regular Domain Account records.

Directory records for Directory-based Domain Accounts must contain the storageLocation attribute. This attribute specifies the location of the Account file directory (for the multi-mailbox accounts) or the location of the Account INBOX (for single-mailbox accounts). The location is specified as a file path relative to the *base directory* of the CommuniGate Pro Server hosting this Account.

If a CommuniGate Pro server has to open an Account in a Directory-based Domain, and the Account storageLocation attribute starts with the asterisk (`*`) symbol, the CommuniGate Pro Server creates the Account file directory (for multi-mailbox Accounts) and other required Account files and file directories.

- If the storageLocation attribute contained only one asterisk, then the new account location path is composed in the same way it is composed for new accounts in the Regular CommuniGate Pro Domains, using the path for the Domain file directory and the Foldering Domain Setting.

Directory records are created for aliases of Directory-based Domain Accounts.
Alias records have the same DNs as Accounts (`uid=aliasname,domain DN`).
Alias records have the standard `alias` objectClass, and their aliasedObjectName attribute specifies the DN of the original account record.

The following diagram illustrates how the LDAP AddRecord operation can be used to create an Account in the Directory-based Domain:
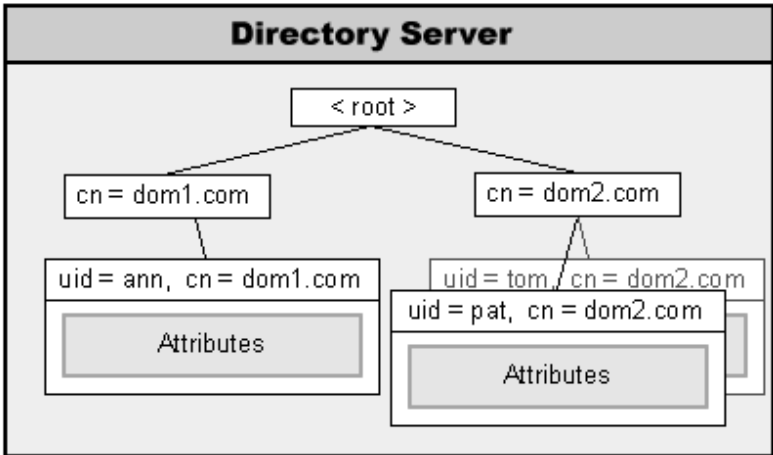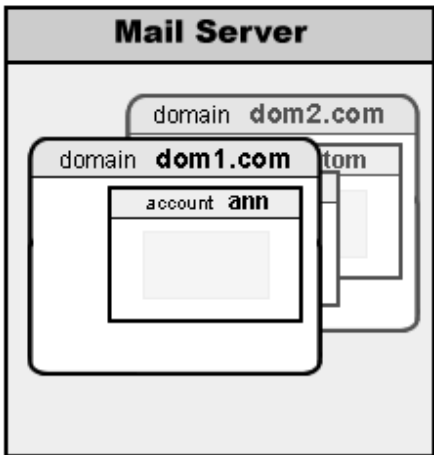
In this example:

- The LDAP module received an AddRecord request from an LDAP client. The client asks the LDAP module to create a new record with the `uid=john,cn=dom1.dom` DN.
- The LDAP module creates a new record in the Directory and stores all supplied attributes in it. The response is sent back to the LDAP client and the operation is completed.
- Any Server component tries to open the account `john` in the Directory-Based Domain `dom1.dom`. The request is sent to the Directory and the record with the `uid=john,cn=dom1.dom` DN is retrieved.
- The storageLocation attribute in the retrieved record contains the asterisk (`*`) symbol. The Server creates the Account files on disk, and updates the Directory record, storing the file path to the Account files in the storageLocation attribute.
- The Account `john` is opened and the requested operation is executed.

Since the Account in the Directory-based Domains do not store their settings in the CommuniGate Pro data files, the settings are retrieved from the account Directory record every time an account has to be opened. The following diagram illustrates this procedure
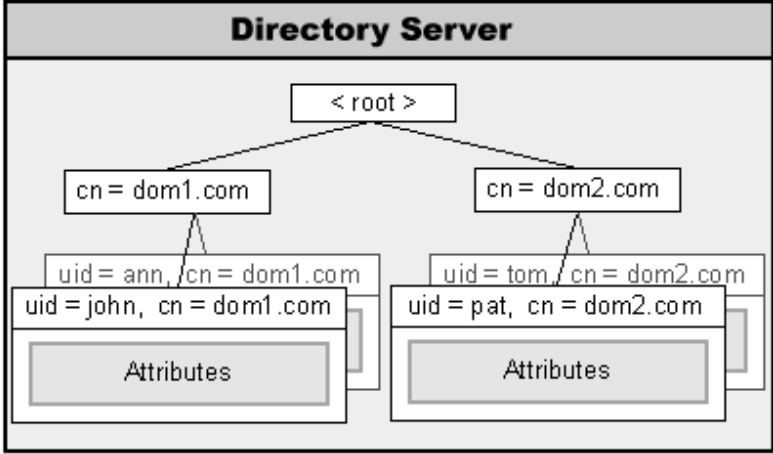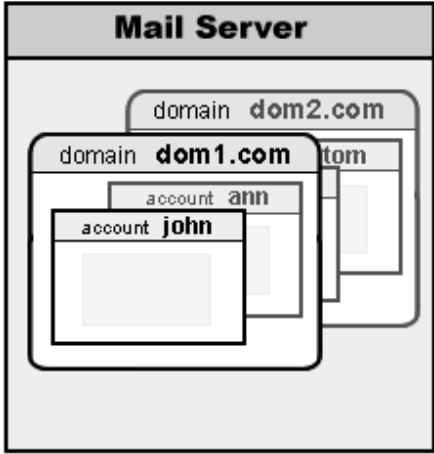


Since the Directory records are the only source of the Account settings, modifying Directory record attributes effectively modifies the Account Settings:
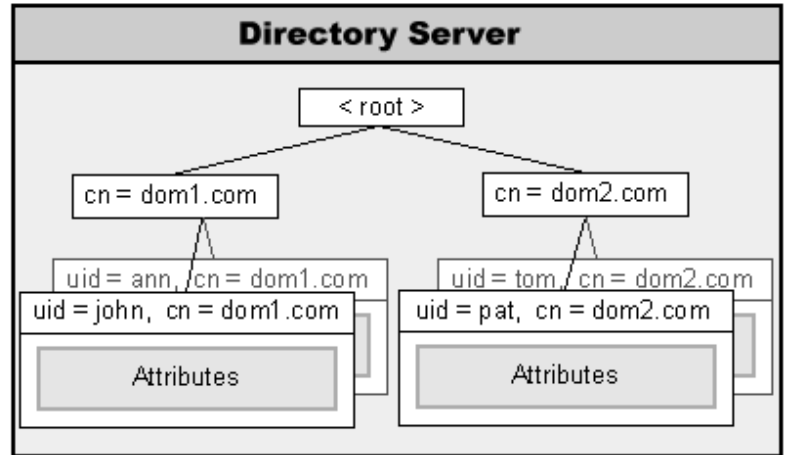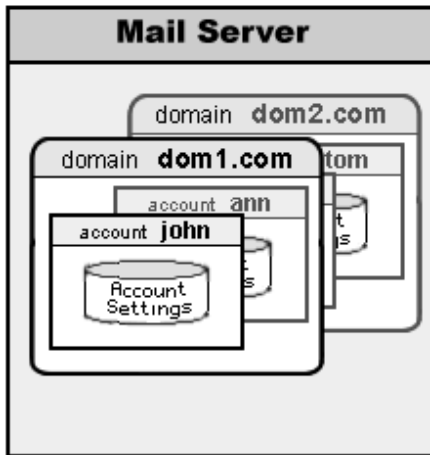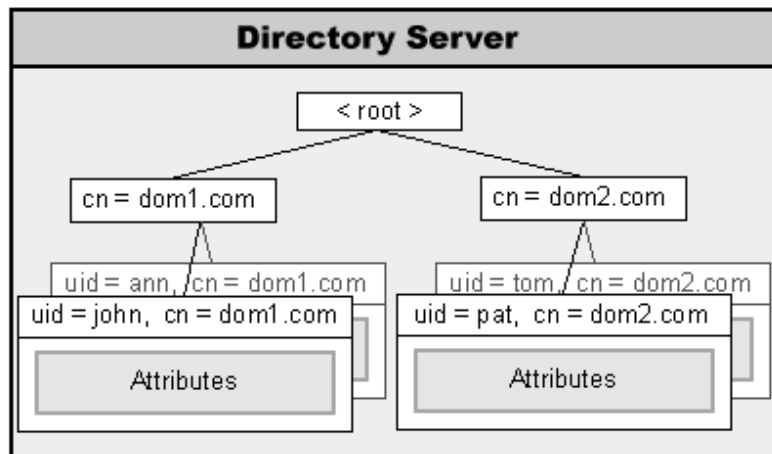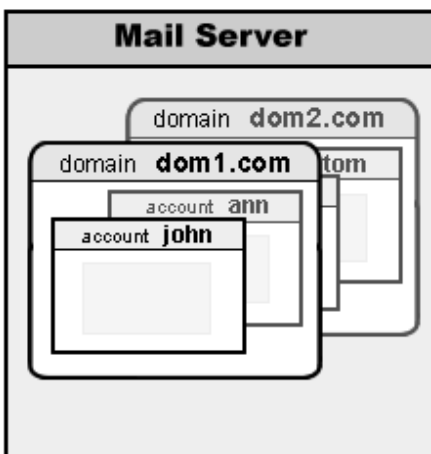
In this example:

- The LDAP module received a ModifyRecord request from an LDAP client. The client asks the LDAP module to change attributes in the `uid=john,cn=dom1.dom` record.
- The LDAP module tells the Directory to modify the specified record. The response is sent back to the LDAP client and the operation is completed.
- Any Server component tries to open the account `john` in the Directory-Based Domain `dom1.dom`. The request is sent to the Directory and the record with the `uid=john,cn=dom1.dom` DN is retrieved and its attributes are used as Account Settings. Since the attribute value has been modified, the Account Setting set with that attribute is effectively modified.

---

# Shared (Multi-Server) Directory

Several CommuniGate Pro Servers can use the same physical Directory (Directory Unit) to keep all their Domain Integration Records.

The shared Directory Unit can be implemented as a Local Storage Unit on one of the CommuniGate Pro Servers, or it can be hosted on some third-party Directory Server.

- Specify the same [Domains Subtree](#) parameters on all CommuniGate Pro Servers.
- On all CommuniGate Pro Server (except the one that will host the shared Directory) create Remote Storage Units for the same Subtrees. The Remote Storage Unit Subtree parameter should be either the same as the Domains Subtree Base DN parameter, or should be its parent, so the entire Domains Subtree will be stored in those Remote Storage Units.
- Configure all those Storage Units to point to the server that will host the shared Directory.

To simplify the setup, especially if you have many CommuniGate Pro Servers, it is recommended to create the Remote Storage Units for the `<root>` Subtrees. To create such a Unit, remove the default `Main` Local Unit first:
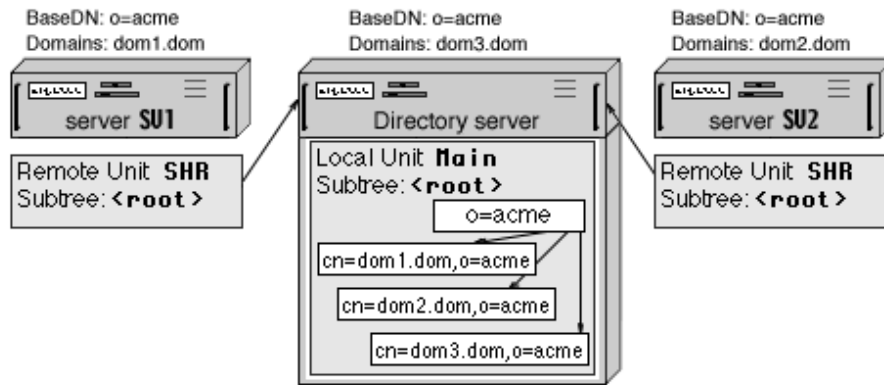
In this example:

- One CommuniGate Pro Server is used to host the Shared Directory.
- The SV1 and SV2 CommuniGate Pro Servers are configured to use that Shared Directory: they both have Remote Storage Units `SHR` for their `<root>` Directory Subtrees, and those Units point to the Directory hosting Server.
- All Servers have the same Directory Integration settings - the Domain Subtree Base DN is `o=acme` for all Servers.
- The actual `o=acme` record is created in the `Main` Local Storage Unit on the Directory Hosting Server.
- The Directory records for:
    - the `dom1.dom` Domain created on the SV1 Server,
    - the `dom2.dom` Domain created on the SV2 Server, and
    - the `dom3.dom` Domain created on the Directory Hosting Server
  are stored in the `Main` Local Storage Unit on the Directory Hosting Server

---

# Distributed Domains (Directory Routing)

When several CommuniGate Pro Servers use a [Shared Directory](#) to keep all their Domain Integration Records, these Servers can be used to serve the same Domain (or the same Domains). Such a Domain is called a Distributed Domain, and each Server hosts a subset of that Domain Accounts. The Distributed Domain should not be a Main Domain of any CommuniGate Pro Server:
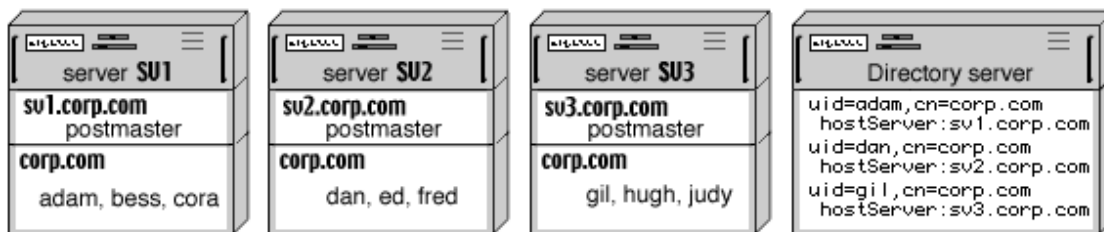


In this example:

- Three CommuniGate Pro Servers (with the `sv1.corp.dom`, `sv2.corp.dom`, and `sv3.corp.dom` Main Domains) all have the same `corp.dom` Secondary Domain. Some Accounts are created in the `corp.dom` Domain on each Server.
- The Domain Subtree Base DN is set to an empty string (`<root>`) on all CommuniGate Pro Servers.
- The Shared Directory is hosted on a separate device/server, but in reality one of the CommuniGate Pro Servers can act as the Shared Directory Host.
- The Directory Integration option of the `corp.dom` Domain is set to `Keep In Sync` on all Servers.

When an Account is created, renamed, removed, or updated on one of the sv*.corp.dom Servers, the Directory Unit on the Shared Directory Server is updated. As a result, the Shared Directory contains records for all Accounts created on all sv*.corp.dom Servers.

When any Server creates an Account and places a record into the Shared Directory, it stores the Server Main Domain

name as the record `hostServer` attribute.

The Shared Directory can be used to route Distributed Domain mail to the proper location (Server). Open the General page in the WebAdmin Settings realm, then open Cluster settings page, and enable the `Directory-Based Clustering`. The address routing mechanism is modified:

- When the CommuniGate Pro Server receives a mail for one of its local Domains, it checks if a local Domain object (Account, Alias, Mailing List, Group, Forwarded) exists.
- If no local object is found in the addressed Domain, the Server checks the Directory.
- If the Directory contains a record for the specified object (`uid=objectName,cn=domainName`), the record `hostServer` attribute is checked.
- If the `hostServer` attribute is absent, or if it contains the Main Domain Name of this CommuniGate Pro Server, an error message is generated. Otherwise, the address is rerouted to the proper relaying module (SIP or SMTP), to the remove server with the *hostserver* name.

This Distributed Domain configuration is useful for multi-location and international organizations and corporations where all employee Accounts should be in the same Domain, but each organizational unit is served with its own Server. The DNS MX records for the such a Distributed Domain should point to any or to all Servers hosting that Domain. When a Server receives mail for a Distributed Domain, it either delivers the mail locally (if the addressed Account is hosted on that Server), or relays mail to Server specified in the hostServer attribute of the Account Directory record.

**Note:** the names and IP addresses for "distributed" systems should be entered into the "Static Members" table on the Clusters WebAdmin page.

Usually, one of the Servers (the "main location") hosts most of the Distributed Domain Accounts. It is recommended to host the Shared Directory on that CommuniGate Pro Server to minimize the delays introduced with the Directory lookups. Other CommuniGate Pro Server serving this Distributed Domain can be configured to reroute all mail to non-local objects of the Distributed Domain to that "main location" Server. Open the Distributed Domain [Settings](#), and set the Mail/Signal To Unknown options to

```
Reroute To: *%domain.dom@mainserver._via
```

This method eliminates a need for "remote location" Servers to communicate with the Directory when they have to route addresses. The "remote location" Servers communicate with the Directory only when Accounts are created in, renamed, or removed from a Distributed Domain, and when a WebMail, XIMSS, SIP, or LDAP user requests a Directory search operation. This can drastically improve the "remote location" Servers performance if the communication links between them and the Shared Directory Server are slow and/or unreliable.

In asymmetric, "main/remote location" configurations, the high-priority MX records for the Distributed Domain should point to the "main location" Server, while "remote location" Server names can be used for low-priority MX records. It is not recommended to use [Directory-based Domains](#) for Distributed Domains if connections between "remote location" Servers and the Shared Directory are slow and/or unreliable.

The Distributed Domains concept is the foundation of the CommuniGate Pro [Static Clusters](#).

For small Distributed Domains, routing can be implemented using regular CommuniGate Pro [Router](#) records. If the Distributed Domain has the same Accounts as shown in the example above, the `SV1` server should have the following records in its Router:

```
; SV2 accounts
<dan@corp.dom>  =  dan%corp.dom@sv2.corp.dom._via
<ed@corp.dom>   =   ed%corp.dom@sv2.corp.dom._via
<fred@corp.dom> = fred%corp.dom@sv2.corp.dom._via
;
; SV3 accounts
<gil@corp.dom>  =  gil%corp.dom@sv3.corp.dom._via
<hugh@corp.dom> = hugh%corp.dom@sv3.corp.dom._via
<judy@corp.dom> = judy%corp.dom@sv3.corp.dom._via
```

While this method does not require any Directory activity, it is hardly acceptable for Domains with more than few dozen Accounts, unless names of Accounts hosted on different Servers can be easily expressed using the Router wildcard symbols. For example, if all Accounts hosted on the Server SV2 end with the `-uk` suffix (`dan-uk@corp.dom`, `ed-uk@corp.dom`, `fred-uk@corp.dom`, etc.), routing for all SV2 Accounts can be specified with one Router record:

```
<*-uk@corp.dom> = *-uk%corp.dom@sv2.corp.dom._via
```

# Directory Integration in a Cluster

The CommuniGate Pro Dynamic Cluster maintains cluster-wide Directory Integration settings: the cluster-wide Attribute Renaming table, Domain Subtree, and Custom Attributes settings are used with all Accounts in Shared Domains, while "regular" settings are used for all Accounts in "local" Domains.

When you open the Directory Integration WebAdmin page on a Cluster Member, the page contains a link that allows you to switch the Cluster-wide Settings.

**Clusters** Static Dynamic Storage E-Mail Real-Time Access Balancer

# Clusters

- **Terminology**
- **Cluster Types**
- **Supported Services**
- **Frontend Servers**
  - **Withdrawing Frontend Servers from a Cluster**
- **Cluster Server Configuration**
  - **Cluster Network**
  - **Cluster Communication**
- **Assigning IP Addresses to Shared Domains**
- **Cluster Configuration Details**
- **Cluster Of Clusters**

When your CommuniGate Pro system should serve more than 150,000-200,000 Accounts, or when you expect a really heavy SIP/IMAP/WebMail/MAPI traffic, you should consider using a multi-server *Cluster configuration*.

# Terminology

If your site serves many Domains, you may want to install several independent CommuniGate Pro Servers and distribute the load by distributing domains between the servers. In this case you do not need to employ the special Cluster Support features. However if you have one or several Domains with 100,000 or more Accounts, and you cannot guarantee that clients will always connect to the proper server, or if you need dynamic load balancing and very high availability, you should implement a CommuniGate Pro Cluster on your site.

Many vendors use the term *Cluster* for simple *fail-over* or *hot stand-by* configurations. The CommuniGate Pro software can be used in fail-over, as well as in Distributed Domains configurations, however these configurations are not referred to as *Cluster configurations*.

A CommuniGate Pro Cluster is a set of Server computers that handle the site mail load together. Each Cluster Server hosts a set of regular, non-shared domains (the CommuniGate Pro Main Domain is always a non-shared one), and it also serves (together with other Cluster Servers) a set of Shared Domains.

To use CommuniGate Pro servers in a Cluster, you need a special CommuniGate Pro Cluster License.

Please read the Scalability section first to learn how to estimate your Server load, and how to get most out of each CommuniGate Pro Server running in the Single-server or in the Cluster mode.

# Cluster Types

There are two main types of Cluster configurations: *Static* and *Dynamic*.

Each Account in a Shared Domain served with a Static Cluster is created (hosted) on a certain Server, and only that Server can access the account data directly. When a Static Cluster Server needs to perform any operation with an account hosted on a different Server, it establishes a TCP/IP connection with the account Host Server and accesses account data via that Host Server. This architecture allows you to use local (i.e. non-shared) storage devices for account data.

**Note:** some vendors have "Mail Multiplexor"-type products. Those products usually implement a subset of Static Cluster Frontend functionality.

Accounts in Shared Domains served with a Dynamic Cluster are stored on a shared storage, so each Cluster Server (except for Frontend Servers, see below) can access the account data directly. At any given moment, one of the Cluster Servers acts as a Cluster Controller synchronizing access to Accounts in Shared Domains. When a Dynamic Cluster Server needs to perform any operation with an account currently opened on a different Server, it establishes a TCP/IP connection with that "current host" Server and accesses account data via that Server. This architecture provides the highest availability (all accounts can be accessed as long as at least one Server is running), and does not require file-locking operations on the storage device.

# Supported Services

The CommuniGate Pro Clustering features support the following services:

- SMTP mail receiving
- SMTP mail delivery
- SIP signaling
- XIMSS access
- WebUser Interface access
- XMPP signaling
- POP3 access
- IMAP access
- FTP
- RADIUS
- TFTP
- HTTP access to File Storage (including uploading)
- ACAP access
- PWD access and remote administration
- CG/PL inter-task communication
- HTTP request to external servers
- RPOP polling
- Remote SIP Registrations
- Cluster-wide Directory Units

# Frontend Servers

Clusters of both types are usually equipped with *Frontend Servers*. Frontend Servers cannot access Account data directly - they always open connections to other (Backend) Servers to perform any operation with Account data.

Frontend servers accept TCP/IP connections from client computers (usually - from the Internet). In a pure Frontend-Backend configuration no Accounts are created on any Frontend Server, but nothing prohibits you from serving some Domains (with Accounts and mailing lists) directly on the Frontend servers.

When a client establishes a connection with one of the Frontend Servers and sends the authentication information (the Account name), the Frontend server detects on which Backend server the addressed Account can be opened, and establishes a connection with that Backend Server.

The Frontend Servers:

- handle all SSL/TLS encryption/decryption operations
- implement SIP Farm functionality, and most of SIP protocol features, including NAT Traversal
- handle most of the SMTP relaying operations themselves
- run Real-Time CG/PL applications and Media Server Channels
- virtually eliminate inter-server communications between Backend Servers, and (in Dynamic Clusters) provide second-level load balancing
- provide an additional layer of protection against Internet attacks and allow you to avoid exposing Backend Servers to the Internet
- smooth out the external traffic (soften peaks in the site load), and protect the Backend Servers from the Denial-of-Service attacks
- execute outgoing HTTP requests.
- execute outgoing Remote SIP Registration tasks.
- execute outgoing Remote POP (RPOP) polling sessions.

If the Frontend Servers are directly exposed to the Internet, and the security of a Frontend Server operating system is compromised so that someone gets unauthorized access to that Server OS, the security of the site is not totally compromised. Frontend Servers do not keep any Account information (Mailboxes, passwords) on their disks. The "cracker" would then have to go through the firewall and break the security of the Backend Server OS in order to get access to any Account information. Since the network between Frontend and Backend Servers can be disabled for all types of communications except the CommuniGate Pro inter-server communications, breaking the Backend Server OS is virtually impossible.

Both Static and Dynamic Clusters can work without dedicated Frontend Servers. This is called a *symmetric configuration*, where each Cluster Server implements both Frontend and Backend functions.

In the example below, the domain1.dom and domain2.dom Domain Accounts are distributed between three Static Cluster Servers, and each Server accepts incoming connections for these Domains. If the Server SV1 receives a connection for the Account `kate@domain1.dom` located on the Server SV2, the Server SV1 starts to operate as a Frontend Server, connecting to the Server SV2 as the Backend Server hosting the addressed Account.
At the same time, an external connection established with the server SV2 can request access to the `ada@domain1.dom` Account located on the Server SV1. The Server SV2 acting as a Frontend Server will open a connection to the Server SV1 and will use it as the Backend Server hosting the addressed Account.

In a symmetric configuration, the number of inter-server connections can be equal to the number of external (user) access-type (POP, IMAP, HTTP) connections. For a symmetric Static Cluster, the average number of inter-server connections is $M*(N-1)/N$, where M is the number of external (user) connections, and the N is the number of Servers in the Static Cluster. For a symmetric Dynamic Cluster, the average number of inter-Server connections is $M*(N-1)/N * A/T$, where T is the total number of Accounts in Shared Domains, and A is the average number of Accounts opened on each Server. For large ISP-type and portal-type sites, the A/T ratio is small (usually - not more than 1:100).

In a pure Frontend-Backend configuration, the number of inter-server connections is usually the same as the number of external (user) connections: for each external connection, a Frontend Server opens a connection to a Backend Server. A small number of inter-server connections can be opened between Backend Servers, too.

# Withdrawing Frontend Servers from a Cluster

To remove a Frontend Server from a Cluster (for maintenance, hardware upgrade, etc.), reconfigure your Load Balancer or the round-robin DNS server to stop redirection of incoming requests to this Frontend Server address. After all current POP, IMAP, SMTP sessions are closed, the Frontend Server can be shut down. Since the WebMail sessions do not use persistent HTTP connections, a Frontend Server in a WebMail-only Cluster can be shut down almost immediately.

Access to all Shared Domain Accounts is provided without interruption as long as at least one Frontend Server is running.

If a Frontend server fails, no Account becomes unavailable and no mail is lost. While POP and IMAP sessions conducted via the failed Frontend server are interrupted, all WebUser Interface session remain active, and WebUser Interface clients can continue to work via remaining Frontend Servers. POP and IMAP users can immediately re-establish their connections via remaining Frontend Servers.

If the failed Frontend server cannot be repaired quickly, its Queue can be processed with a different server, as a Foreign Queue.

# Cluster Server Configuration

This section specifies how each CommuniGate Pro Server should be configured to participate in a Static or Dynamic Cluster. These settings control inter-server communications in your Cluster.

First, install CommuniGate Pro Software on all Servers that will take part in your Cluster. Specify the Main Domain Name for all Cluster Servers. Those names should differ in the first domain name element only:

> `back1.isp.dom`, `back2.isp.dom`, `front1.isp.dom`, `front2.isp.dom`, etc.

Remember that Main Domains are never shared, so all these names should be different. You may want to create only the Server administrator accounts in the Main Domains - these accounts can be used to connect to that particular Server and configure its local, Server-specific settings.

## Cluster Network

Use the WebAdmin Interface to open the Settings->General->Cluster page on each Backend and Frontend Server, and enter all Frontend and Backend Server IP addresses. CommuniGate Pro Servers will accept Cluster connections from the specified IP addresses only. If the Frontend Servers use dedicated Network Interface Cards (NICs) to communicate with Backend Servers, specify the IP addresses the Frontend Servers have on that internal network:

| Cluster Network | | |
| --- | --- | --- |
| | On restart | Active |
| This Server Cluster Address: | [192.168.0.5] | [192.168.0.5] |
| Backend Server Addresses: | | |
| | | |
| Frontend Server Addresses: | | |

`This Server Cluster Address`

This setting specifies the local network address this Server will use to communicate with other Servers in the Cluster. Connections to other Servers will be established from this IP address. This address is used as this Server "name", identifying the Server in the Cluster.

If you change this setting value, the new value will be in effect only after Server restart.

# Cluster Communication

In all types of CommuniGate Pro cluster, connections to Backend servers can be established from Frontend servers and from other Backend servers.

If your Backend Servers use non-standard port numbers for its services, change the Backend Server Ports values.

For example, if your Backend Servers accept [WebUser Interface](#) connections not on the port number 8100, but on the standard HTTP port 80, set 80 in the `HTTP User` field and click the Update button.

| Inter-Cluster Communications | | | |
|---|---|---|---|
| **Backend Port** | **Cache** | **Backend Port** | **Cache** |
| Delivery: | 30 | Submit: | 5 |
| POP: | | IMAP: | |
| HTTPU: | | HTTPA: | |
| XIMSS: | | XMPP: | |
| PWD: | | ACAP: | |
| LDAP: | | | |
| **Log Level** | **Cache** | **Log Level** | **Cache** |
| Object Admin: Problems | 10 | Mailboxes: Low Level | 10 |
| Cluster Admin: Problems | | | |

CommuniGate Pro can reuse inter-server connections for some services. Instead of closing a connection when an operation is completed, the connection is placed into an internal cache, and it is reused later, when this server needs to connect to the same server. The Cache parameter specifies the size of that connection cache. If there are too many connections in the cache, older connection are closed and pushed out of the cache.

Cluster members use the PWD protocol to perform administration operations remotely, on other cluster members. The port number they use to connect to on other Cluster members is the same as the port specified for the PWD protocol connections. These remote administrative operations have their own Log Level settings.

Servers in a Dynamic Cluster use the SMTP modules of other Cluster Backend members for remote message delivery (though the protocol between the servers is not the SMTP protocol). Use the `Delivery` port setting to specify the port number used with SMTP modules on other cluster members.

Servers in a Dynamic Cluster use the SMTP modules of other Cluster members to submit messages remotely (though the protocol between the servers is not the SMTP protocol). Use the `Submit` port setting to specify the port number

used with SMTP modules on other cluster members.

When a user session running on one Cluster member needs to access a *foreign Mailbox*, and the account that this Mailbox belongs to cannot be opened by the same Cluster member, the Cluster Mailbox manager is used to access Mailboxes remotely. The Cluster Mailbox manager uses the IMAP port to connect to other cluster members. The Cluster Mailbox manager has its own Log Level setting.

| Service Processing | | | |
|---|---|---|---|
| HTTP Client: | Auto | RPOP Client: | Auto |

When a Cluster is configured so that only the frontend servers can access the Internet, certain services can run on those frontend servers only.

HTTP Client
>This setting specifies how the outgoing HTTP requests (initiated with [XIMSS](#) sessions, [CG/PL](#) applications, [Automated Rules](#), etc.) are executed.

>Locally
>>when there is an HTTP request to execute, it is executed on the same Server (this is the "regular", single-server processing mode).

>Locally for Others
>>HTTP requests are executed on the same Server.
>>The Dynamic Cluster Controller is informed that this Server can execute HTTP requests for other Cluster members.
>>The Dynamic Cluster Controller collects and distributes information about all active Cluster members that have this option selected.

>Remotely
>>when there is an HTTP request to execute, a request is relayed to some Cluster member that has this setting set to Locally for Others.

>Auto

>>- if this Server is not a Dynamic Cluster member, same as Locally
>>- if this Server is a Dynamic Cluster frontend, same as Locally for Others
>>- if this Server is a Dynamic Cluster backend, same as Remotely if there are other Dynamic Cluster members configured as Locally for Others, if there are none - same as Locally

RPOP Client
>This setting specifies how the outgoing [RPOP](#) requests are executed. The setting values have the same meanings as the HTTP Client setting values.

---

# Assigning IP Addresses to Shared Domains

A CommuniGate Pro Cluster can serve several Shared Domains. If you plan to provide POP and IMAP access to Accounts in those Domains, you may want to assign dedicated IP addresses to those Domains to simplify client mailer setups. See the [Access](#) section for more details.

If you use Frontend Servers, only Frontend Servers should have dedicated IP Addresses for Shared Domains. Inter-server communications always use full account names (*accountname@domainname*), so there is no need to dedicate IP

Addresses to Shared Domains on Backend Servers.

If you use the DNS round-robin mechanisms to distribute the site load, you need to assign N IP addresses to each Shared Domain that needs dedicated IP addresses, where N is the number of your Frontend Servers. Configure the DNS Server to return these addresses in the round-robin manner:
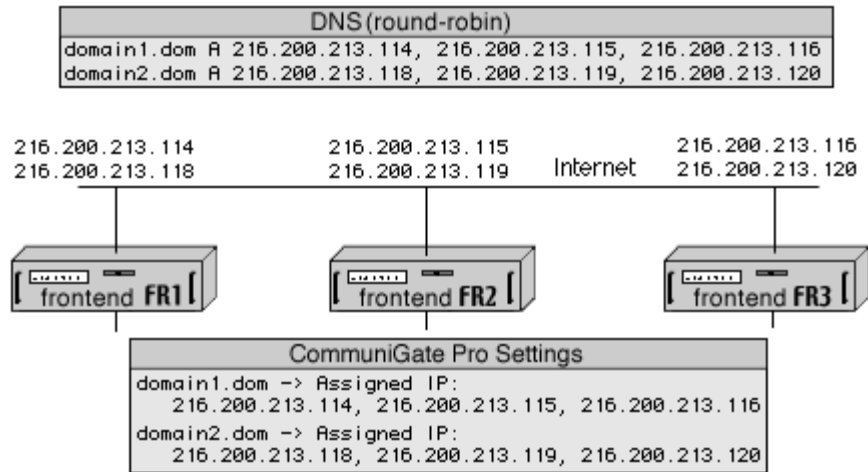


In this example, the Cluster is serving two Shared Domains: domain1.dom and domain2.dom, and the Cluster has three Frontend Servers. Three IP addresses are assigned to each domain name in the DNS server tables, and the DNS server returns all three addresses when a client is requesting A-records for one of these domain names. Each time the DNS server "rotates" the order of the IP addresses in its responses, implementing the DNS "round-robin" load balancing (client applications usually use the first address in the DNS server response, and use other addresses only if an attempt to establish a TCP/IP connection with the first address fails).

When configuring these Shared Domains in your CommuniGate Pro Servers, you assign all three IP addresses to each Domain.

If you use a Load Balancer to distribute the site load, you need to place only one "external" IP address into DNS records describing each Shared Domain. You assign one "virtual" (LAN) IP address to each Shared Domain on each Frontend Server:
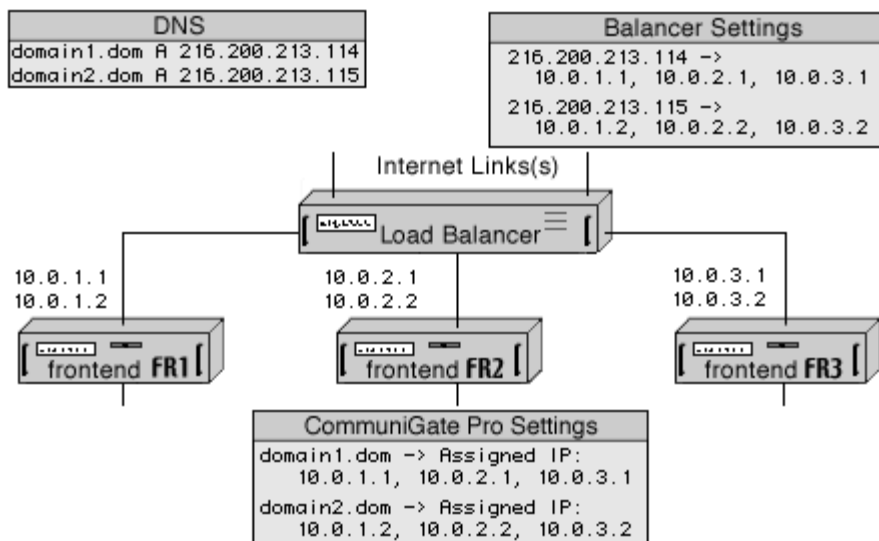


In this example, the Cluster is serving two Shared Domains: domain1.dom and domain2.dom, and the Cluster has three Frontend Servers. One IP Addresses assigned to each Shared Domain in the DNS server tables, and those addresses are external (Internet) addresses of your Load Balancer. You should instruct the Load Balancer to distribute connections received on each of its external IP addresses to three internal IP addresses - the addresses assigned to your

Frontend Servers.

When configuring these Shared Domains in your CommuniGate Pro Servers, you assign these three internal IP addresses to each Domain.

DNS MX-records for Shared Domains can point to their A-records.

---

# Cluster Configuration Details

## Startup Parameters

The best way to specify additional startup parameters is to create the `Startup.sh` file inside the *base directory*.

To create a Static Cluster, use the `--staticBackend` startup parameter with the Backend servers, and the `--staticFrontend` startup parameter with the Frontend servers.

To create a Dynamic Cluster, use the `--clusterBackend` startup parameter with the Backend servers, and the `--clusterFrontend` startup parameter with the Frontend servers.

## Listeners

To protect your site from DoS attacks, you may want to open SMTP, POP, IMAP, and other Listeners and limit the number of connections accepted from the same IP address. Set those limits on Frontend servers only, since Backend servers receive all connections from Frontends, and each Frontend can open a lot of connections from the same IP address.

## WebAdmin

Usually the Backend servers are not directly accessible from the Internet. If you need to change the settings or monitor one of the Backend servers from "outside", you can use the WebAdmin interface of one of the Frontend servers, using the following URL:
`http://Frontendaddress:8010/Cluster/12.34.56.78/`
where 12.34.56.78 is the [internal] IP address of the Backend server you want to access.

## SMTP

The outgoing mail traffic generated with regular (POP/IMAP) clients is submitted to the site using the A-records of the site Domains. As a result, the submitted messages go to the Frontend Servers and the messages are distributed from there.

Messages generated with WebUser clients and messages generated automatically (using the Automated Rules) are generated on the Backend Servers. Since usually the Backend servers are behind the firewall and since you usually do not want the Backend Servers to spend their resources maintaining SMTP queues, it is recommended to use the forwarding feature of the CommuniGate Pro SMTP module.

Select the Forward to option and specify the asterisk (`*`) symbol. In this case all messages generated on the Backend Servers will be quickly sent to the Frontend Servers and they will be distributed from there. If you do not want to use all Frontend servers for Backend mail relaying, change the Forward To setting to include the IP addresses of some Frontend Servers, separating the addresses with the comma (`,`) symbol.

## RPOP

In a Static Cluster, RPOP activity takes place on Backend servers. As a result, it is essential for those servers to be able to initiate outgoing TCP connections to remote servers. If the Backend servers are connected to a private LAN behind a firewall, you should install some NAT server software on that network and configure the Backend servers (using their OS TCP/IP settings) to route all non-local packets via the NAT server(s). Frontend servers can be used to run NAT services.

In a Dymanic Cluster, RPOP activity takes place on those servers that have the RPOP service set to Locally for Others. These servers (usually - frontends) should be able to initiate outgoing TCP connections to remote servers. RPOP activity is scheduled on the active Cluster Controller, so if Backend servers do not have direct access to the Internet, their RPOP setting should be set to Remotely.

## FTP

The FTP module does not "proxy" connections to Backend servers. Instead, it uses CLI to manage Account File Storage data on Backend servers. This eliminates a problem of Backend servers opening FTP connections directly to clients. If all FTP connections come to the Frontend servers, the FTP services on Backends can be switched off.

The FTP module running on cluster Frontends behind a load balancer and/or a NAT has the same problems as any FTP server running in such a configuration. To support the active mode, make sure that Frontend servers can open outgoing connections to client FTP ports (when running via a NAT, make sure that the "address in use" problems are addressed by the NAT software). To support the passive mode, make sure that your load balancer allows clients to connect directly to the Frontend ports the FTP module opens for individual clients.

## LDAP

The LDAP module does not "proxy" connections to Backend servers. Instead, it uses CLI to authenticate users and, optionally, to perform LDAP Provisioning operations. If all LDAP connections come to the Frontend servers, the LDAP services on Backends can be switched off, unless they serve Cluster-wide Directory Units (see below).

## Directory

Cluster-wide Directory Units are processed on the Cluster Controller server. Other Cluster members communicate with Cluster-wide Units by sending LDAP requests to the Controller.
If you use Cluster-wide Directory Units, the LDAP service on the Backend servers should be enabled.

## RADIUS

The RADIUS module does not "proxy" connections to Backend servers. Instead, it uses CLI to authenticate users and to update their statistical data. If all RADIUS connections come to the Frontend servers, the RADIUS services on Backends can be switched off.

## SIP

See the Cluster Signals section.

## RSIP

RSIP processing is similar to RPOP processing. RSIP registrations are implemented as Real-Time Tasks, so they run on those servers that can run Call Legs. See the Cluster Signals section.

## Postmaster Account

Do not remove the "`postmaster`" Account from the Main Domains on your Backend servers. This Account is opened "by name" (bypassing the Router) when any other Cluster member has to connect to that Backend. You should also keep at least the [All Domains](#) access right for the `postmaster` Account.

# Cluster Of Clusters

For extremely large sites (more than 5,000,000 active accounts), you can deploy a Static Cluster of Dynamic Clusters. It is essentially the same as a regular Static Cluster with Frontend Servers, but instead of Backend Servers you install Dynamic Clusters. This solves the redundancy problem of Static Clusters, but does not require extremely large Shared Storage devices and excessive network traffic of extra-large Dynamic Clusters:
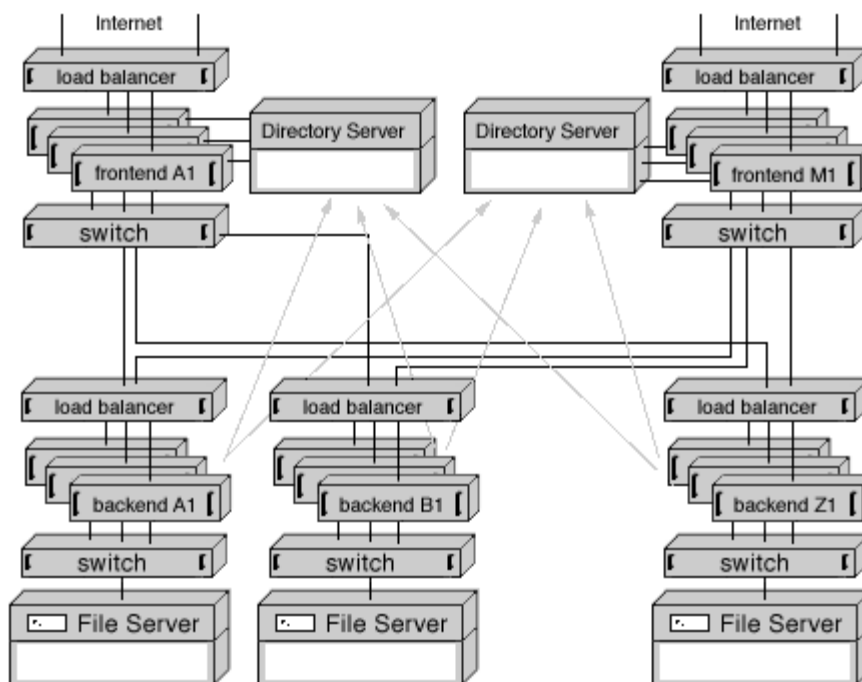


Frontend Servers in a "Cluster of Clusters" need access to the Directory in order to implement Static Clustering. The Frontend Servers only read the information from the Directory, while the Backend Servers modify the Directory when accounts are added, renamed, or removed. The `hostServer` attribute of Account directory records contains the name of the Backend Dynamic Cluster hosting the Account (the name of Backend Cluster Servers without the first domain name element).

Frontend Servers can be grouped into subsets for traffic segmentation. Each subset can have its own load balancer(s), and a switch that connects this Frontend Subset with every Backend Dynamic Cluster.

If you plan to deploy many (50 and more) Frontend Servers, the Directory Server itself can become the main site bottleneck. To remove this bottleneck and to provide redundancy on the Directory level, you can deploy several Directory Servers (each Server serving one or several Frontend subsets). Backend Dynamic Clusters can be configured to update only one "Master" Directory Server, and other Directory Servers can use replication mechanisms to synchronize with the Master Directory Server, or the Backend Clusters can be configured to modify all Directory Servers at once.

Clusters **Static** Dynamic Storage E-Mail Real-Time Access Balancer

# Static Clusters

- **Shared Domains**
- **Backend and Frontend Server Settings**
- **Adding Servers to a Static Cluster**
- **Withdrawing Servers from a Static Cluster**
- **Backend Failover in a Static Cluster**

Static Clusters can be used to handle extremely large (practically unlimited) sites, providing 24x7 site access.

Static Clusters are *loosely-coupled* systems: each Server works almost independently of the other Servers. The Static Cluster setup is an extension of the CommuniGate Pro Distributed Domains configuration.

If a Backend Server fails, the Static Cluster continues to operate, and access to Accounts on the failed Server can be restored within 2-10 minutes (depending on how easily the disk storage can be reassigned and how fast the Routing tables/Directory can be updated, or how quickly a stand-by Server can be switched on).

## Shared Domains

Shared Domains in a Static Cluster are created in the same way as regular CommuniGate Pro Domains. Each Server in a Static Cluster contains a subset of all Shared Domain Accounts. As a result, each Shared Domain Account has its "Host Server". Only the Host Server needs physical access to the Account data, so Static Clusters can use regular, non-shared disk storage. Static Clusters rely on some method that allows each Cluster Server to learn the name of the Host Server for any Shared Domain account. This type of routing can be implemented using a shared Directory Server, in the same way it is implemented for Distributed Domains:
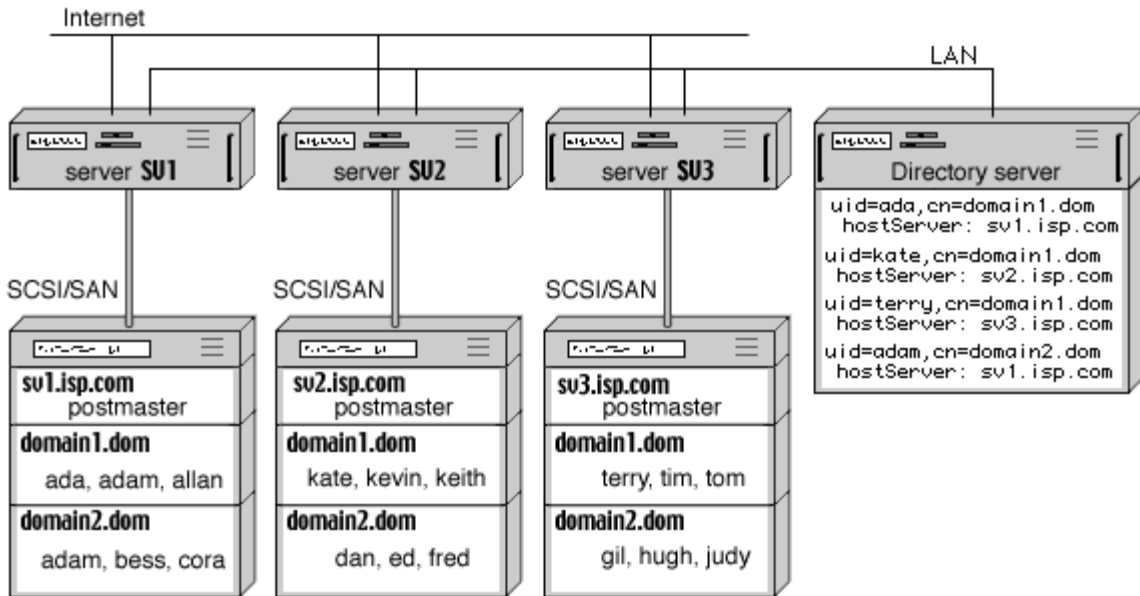
# Backend and Frontend Server Settings

To set a Static Cluster:

- Install and [configure](#) CommuniGate Pro Software on all Servers that will take part in a Static Cluster.
- Configure all Servers to use one [Shared Directory](#) for all Shared Domains.
- Create Shared Domains on all (Backend and Frontend) Servers in the same way regular, non-shared Domains are created.
- Use the WebAdmin Interface to open the Settings->General->Cluster page on each Server, and enter the names (Main Domain Names) of all Backend Servers and the IP addresses of those Servers.

| Static Clustering | |
|---|---|
| **Member Name** | **Member Address** |
| | |

If an address is routed to a domain listed in this table, the CommuniGate Pro Server uses its Clustering mechanism to connect to the Backend server at the specified address and performs the requested operations on that Backend server.

The logical setup of the Backend and Frontend Servers is the same - you simply do not create Shared Domain Accounts on any Frontend Server, but create them on your Backend Servers.

Computers in a Static Cluster can use different operating systems.

A complete Frontend-Backend Static Cluster configuration uses Load Balancers and several separate networks:
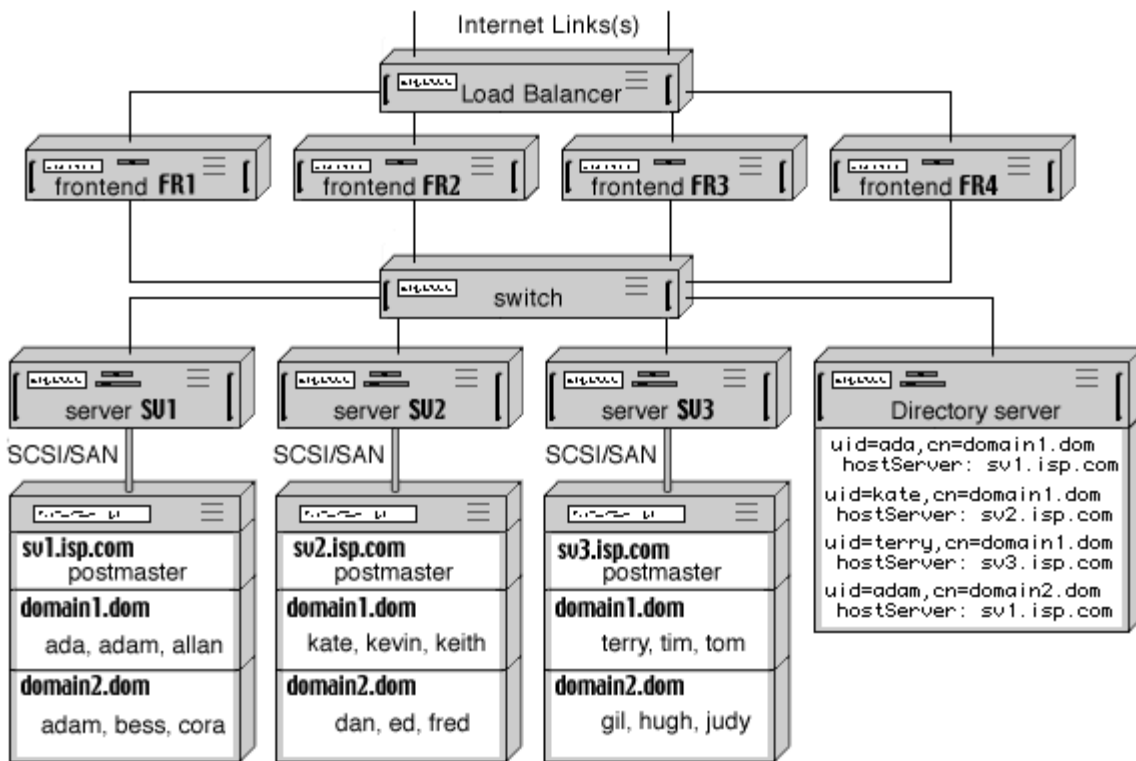
In a simplified configuration, you can connect Frontend Servers directly to the Internet, and balance the load using the DNS *round-robin* mechanism. In this case, it is highly recommended to install a firewall between Frontend and Backend Servers.

---

# Adding Servers to a Static Cluster

You can add Frontend and Backend Servers to a Static Cluster at any time.

To add a Server to a Static Cluster:

- Properly configure the Server (see above): configure it to access the Shared Directory, create Shared Domains, and set the Clustering Settings.
- Add the IP address of the new Server to the Backend or Frontend Addresses tables of other Cluster Members (if you have specified proper network address ranges for those tables, this step is not needed).
- If the new Server is a Backend one, add its name and IP Address to the Static Clustering tables on other Servers.

After a new Frontend Server is configured and added to the Static Cluster, reconfigure the Load Balancer or the round-robin DNS server to direct incoming requests to the new Server, too.

After a new Backend Server is configured and added to the Static Cluster, you can start creating Accounts in its Shared Domains.

---

# Withdrawing a Server from a Static Cluster

If you decide to shut down a Static Cluster Backend Server, all Accounts hosted on that Server become unavailable. Incoming messages to unavailable Accounts will be collected in the Frontend Server queues, and they will be delivered as soon as the Backend Server is added back or these Accounts become available on a different Backend Server (see below).

# Backend Failover in a Static Cluster

If a Backend Server in a Static Cluster is shut down, all Accounts hosted on that Server become unavailable (there is no interrupt in service for Accounts hosted on other Backend Servers).

To restore access to the Accounts hosted on the failed Server, its Account Storage should be connected to any other Backend server. You can either:

* physically connect the disk storage to some other Backend Server;
* use dual-access RAID devices and tell the sibling Server to take over that device;
* use a file server partition or file directory for each Backend Account Storage, and mount that directory on some other Backend Server in case of a Backend Server failure.

After a sibling Backend server gets physical access to Account Storage of the failed server, you should modify the Directory so all Servers will contact the new "home" for Accounts in that Storage. This can be done by an LDAP utility that modifies all records in the Domains Subtree that contain the name of the failed Server as the `hostServer` attribute value. The utility should set the attribute value to the name of the new Host Server, and should add the `oldHostServer` attribute with the name of the original Host Server. This additional attribute will allow you to restore the `hostServer` attribute value after the original Host Server is restored and the Account Storage is reconnected to it. If the CommuniGate Pro is used as the site Directory Server, 500,000 Directory records can be modified within 1-2 minutes.

**Clusters** **Static** **Dynamic** **Storage** **E-Mail** **Real-Time** **Access** **Balancer**

# Dynamic Clusters

- **Traditional File-Lock Approach**
- **Cluster Controller**
- **Cluster File Systems and Cluster OSes**
- **Configuring Backend Servers**
- **Adding a Backend Server to a Dynamic Cluster**
- **Adding a Frontend Server to a Dynamic Cluster**
- **Shared Settings**
- **Shared Processing**
- **Withdrawing Servers from a Dynamic Cluster**
- **Upgrading Servers in a Dynamic Cluster**

While Static Clusters can be used to handle very large sites, they do not meet the *carrier-grade* uptime requirements.

Managing a set of loosely-coupled Server also becomes a problem as the number of Server grows.

The CommuniGate Pro Dynamic Clusters address these challenges. They exceed the "five-nine" (99.999%) uptime requirements, and their Single Service Image infrastructure allows System and Domain administrators manage a large Cluster System in the same way a smaller single-server CommuniGate Pro system is managed.

The main difference between Static and Dynamic Clusters is the Account hosting. While each Account in a Static Cluster has its Host Server, and only that Server can access the Account data directly, all Backend Servers in a Dynamic Cluster can access the Account data directly.

The most common method to implement a Dynamic Cluster shared Account Storage is to employ File Servers or Cluster File Systems. See the Storage section for more information about Shared File Systems.

# Traditional File-Locking Approach

Many legacy Communication servers can employ file servers for account data storage. Since those servers are usually implemented as multi-process systems (under Unix), they use the same synchronization methods in both single-server and multi-server environments, such as *file locks* implemented on the Operating System/File System level.

This method has the following problems:

- Every operation with Account/Mailbox data should be surrounded with file locking/unlocking operations, and additional File System operations are needed to ensure data consistency. As a result, the number of File System operations increases in 3-5 times, and (since the speed of file operation usually defines the speed of the site) the site performance suffers a lot.
- Modern File Servers either do not support file locking mechanisms at all, or provide severely limited versions of

those mechanisms, making the most important site component - account storage - unreliable and not fault-tolerant.

- Malfunction of one of the servers can bring the entire site down (because of deadlocks), and makes fault recovery extremely painful.
- Simultaneous access to the same Account/Mailbox by several clients is either prohibited or unreliable.

In the attempt to decrease the negative effect of file-locking, some legacy Messaging servers support the MailDir Mailbox format only (one file per message), and they rely on the "atomic" nature of file directory operations (rather than on file-level locks). This approach theoretically can solve some of the outlined problems (in real-life implementations it hardly solves any), but it results in wasting most of the file server storage, and overloads the file server internal filesystem tables.
The performance of File Servers severely declines when an application uses many smaller files instead of few larger files.

While simple clustering based on Operating System/File System multi-access capabilities works fine for Web servers (where the data is not modified too often), it does not work well for Messaging servers where the data modification traffic is almost the same as the data retrieval traffic.

Simple Clustering does not provide any additional value (like Single Service Image), so administering a 30-Server cluster is even more difficult than administering 30 independent Servers.

The CommuniGate Pro software supports the [Legacy INBOX](#) feature, so a file-based clustering can be implemented with the CommuniGate Pro, too. But because of the problems outlined above, it is highly recommended to avoid this type of solutions and use the real CommuniGate Pro Dynamic Cluster instead.

# Cluster Controller

CommuniGate Pro Servers in a Dynamic Cluster do not use Operating System/File System locks to synchronize Account access operations. Like in a Static Cluster, only one Server in a Dynamic Cluster has direct access to any given Account at any given moment. All other Servers work through that Server if they want to access the same Account. But this assignment is not static: any Server can open any Account directly if that Account is not opened with some other Server.

This architecture provides the maximum uptime: if a Backend Server fails, all Accounts can be accessed via other Backend Servers - without any manual operator intervention, and without any downtime. The site continues to operate and provide access to all its Accounts as long as at least one Backend Server is running.

One of the Backend Servers in a Dynamic Cluster acts as the *Cluster Controller*. It synchronizes all other Servers in the Cluster and executes operations such as creating Shared Domains, creating and removing accounts in the shared domains, etc. The Cluster Controller also provides the *Single Service Image* functionality: not only a site user, but also a site administrator can connect to any Server in the Dynamic Cluster and perform any Account operation (even if the Account is currently opened on a different Server), as well as any Domain-level operations (like Domain Settings modification), and all modifications will be automatically propagated to all Cluster Servers.

**Note:** most of the Domain-level update operations, such as updating Domain Settings, Default Account Settings, WebUser Interface Settings, and Domain-Level Alerts may take up to 30 seconds to propagate to all Servers in the Cluster. Account-Level modifications come into effect on all Servers immediately.

The Cluster Controller collects the load level information from the Backend Servers. When a Frontend Server receives a session request for an Account not currently opened on any Backend Server, the Controller directs the Frontend Server to the least loaded Backend Server. This second-level load balancing for Backend Server is based on actual load levels and it supplements the basic first-level Frontend load balancing (DNS round-robin or traffic-based).

When a Dynamic Cluster has at least 2 backend Servers, the Cluster Controller assigns the Controller Backup duties to one of the other backend Servers. All other Cluster members maintain connections with the Backup Controller. If the Backup Controller fails, some other backend Server is selected as a Backup Controller.

If the main Controller fails, the Backup Controller becomes the Cluster Controller. All Servers send the resynchronization information to the Backup Controller and the Cluster continues to operate without interruption.

While the Dynamic Cluster can maintain a Directory with Account records, the Dynamic Cluster functionality does not rely on the Directory. If the Directory is used, it should be implemented as a [Shared Directory](#).

A complete Frontend-Backend Dynamic Cluster configuration uses Load Balancers and several separate networks:
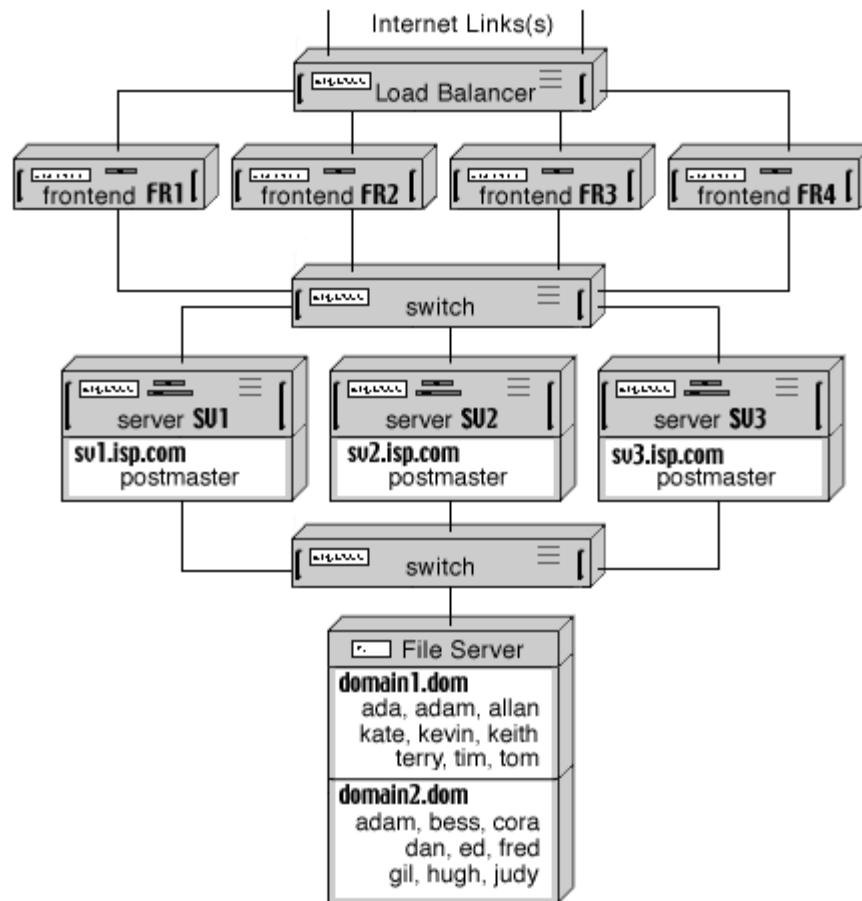


Since all Backend Servers in a Dynamic Cluster have direct access to Account data, they should run the operating systems using the same EOL (end-of-line) conventions. This means that all Backend Servers should either run the same or different flavors of the Unix OS, or they all should run the same or different flavors of the MS Windows OS. Frontend Servers do not have direct access to the Account data, so you can use any OS for your Frontend Servers (for example, a site can use some Unix OS for Backend Servers and Microsoft Windows for Frontend Servers).

---

# Cluster File Systems and Cluster OSes

Some of the modern Operating Systems provide advanced Clustering capabilities themselves. Most of those Cluster features are designed to help porting "regular", non-clustered applications on these Cluster platforms. But some features provided with those Cluster OSes are very useful for the CommuniGate Pro Dynamic Cluster implementations.

These features include:

- Cluster File System
- IP Aliasing

A Cluster File System allows all Servers in an OS Cluster to mount and use the same file system(s) on shared devices. Unlike Network File Systems (NFS), Cluster File Systems do not require a dedicated server on the network. Cluster File Systems can utilize multiple SCSI connections provided with some high-end SCSI storage devices, and they can allow each Server to exchange the data directly with storage devices via a SAN (Storage Area Network). To ensure file system integrity, Cluster File Systems use high-speed server interconnects.

The SAN protocols are very effective for file transfers, and Cluster File Systems can provide better performance than Network File Systems.
The Cluster File Systems can also provide better reliability than single-server NFS solutions (where the NFS server is a single point of failure).
See the Storage section for more details.

The IP Aliasing feature allows the Cluster OS to distribute the network load between Cluster Servers without an additional Load Balancer unit.

A "backend-only" CommuniGate Pro Dynamic Cluster can utilize both features of a Cluster OS: the IP Aliasing is used to distribute the load between CommuniGate Pro Server, and CommuniGate Pro Servers use the Cluster File System to store all account data in shared Domains:
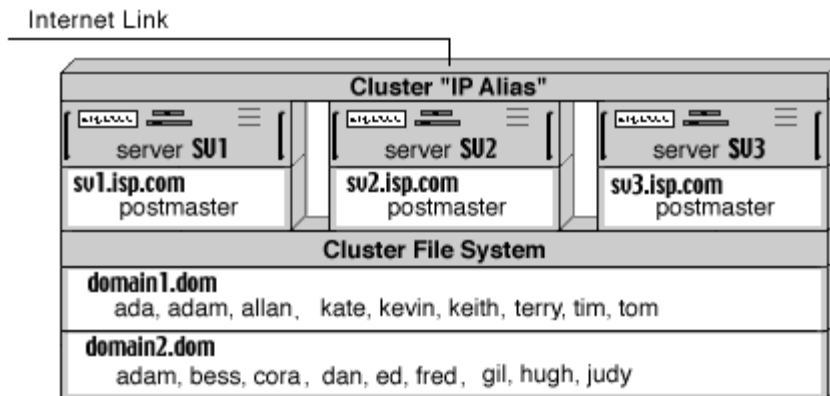


A Cluster OS can be used in a frontend/backend CommuniGate Pro Cluster configuration, too. In this case, one OS Cluster is used for CommuniGate Pro frontend Servers, utilizing the IP Aliasing load balancing, and the second OS Cluster is used for CommuniGate Pro backend Servers, where the Cluster File System is employed:



763
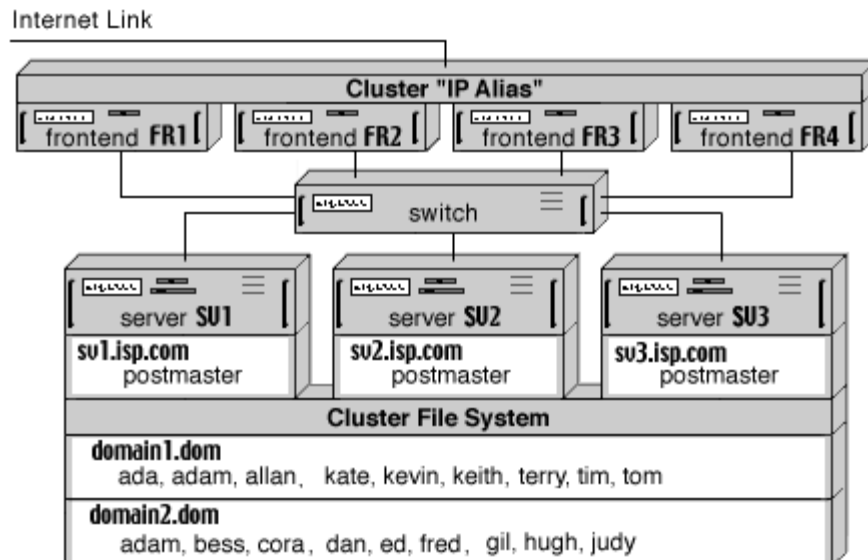
The Configuration of the CommuniGate Pro Dynamic Cluster does not depend on the type of the load balancing used (separate Load Balancers or IP Aliases), or on the type of the shared file system used (Network File System or Cluster File System).

---

# Configuring Backend Servers

To install a Dynamic Cluster, follow these steps:

- Install and configure CommuniGate Pro Software on all Servers that will take part in a Dynamic Cluster.
- Open the WebAdmin Settings->Access page and modify the PWD service settings. Each Cluster member (Backend and Frontend) opens 2 PWD connections to the Cluster Controller, so the maximum number of channels should be increased at least by

      2*(number of Backend servers + number of Frontend servers)

  Since additional PWD connections can be opened by Frontend and Backend servers to serve administrator and user requests, it is better to increase the number of channels by:

      5*(number of Backend servers) + 3*(number of Frontend servers)

- Open the WebAdmin Settings->General->Clusters page and enter the IP addresses of all backend and frontend Servers in the Cluster.
- Stop all Servers.
- Create a file directory that will contain Shared Domains. You should create that file directory on a storage unit that will be available for all Cluster Backend Servers (on a file server, for example). Place a link to that directory into the CommuniGate Pro *base directory*, and name that link `SharedDomains`. Make sure that all Backend Servers have all file access rights to create, remove, read, and modify files and directories inside the SharedDomains directory.

  **Note:** if creating symbolic links is problematic (as it is on MS Windows platforms), you should specify the location of the "mounted" file directory as the `--SharedBase` Command Line Option:

      --SharedBase H:\Base

- If you are upgrading from a single-server configuration, you may want to make some of your existing Domain shared, so they will be served with the entire Cluster. In this case you should move the Domain file directory from the `{base}/Domains` file directory into the `{base}/SharedDomains` file directory (located on a shared storage unit).
- Modify the Startup option for all your Backend Server, so they will include the `--ClusterBackend` Command Line Option.
- Start one of the Backend Servers.

Use the WebAdmin Interface of this first Backend Server to verify that the Cluster Controller is running. Open the Domains page to check that:

- all domains you have placed into the SharedDomains directory are visible;
- the Create Domain button is now accompanied with the Create Shared Domain button.

Use the Create Shared Domain button to create additional Shared Domains to be served with the Dynamic Cluster.

When the Cluster Controller is running, the site can start serving clients (if you do not use Frontend Servers). If your configuration employs Frontend servers, at least one Frontend Server should be started.

---

# Adding a Backend Server to a Dynamic Cluster

Additional Backend Server can be added to the Cluster at any moment. They should be pre-configured in the exactly the same way as the first Backend Server was configured.

To add a Backend Server to your Dynamic Cluster, start it with the `--ClusterBackend` Command Line option (it can be added to the CommuniGate Pro startup script). The Server will poll all specified Backend Server IP Addresses until it finds the active Cluster Controller.

Use the WebAdmin interface to verify that the Backend Server is running. Use the Domains page to check that all Shared Domains are visible and that you can administer Accounts in the Shared Domains.

When the Cluster Controller and at least one Backend Server are running, they both can serve all accounts in the Shared Domains. If you do not use Frontend Servers, load-balancing should be implemented using a regular load-balancer switch, DNS round-robin, or similar technique that distributes incoming requests between all Backend Servers.

---

# Adding a Frontend Server to a Dynamic Cluster

You can add additional Frontend servers to the Cluster at any moment.

Install and Configure the CommuniGate Pro software on a Frontend Server computer. Since Frontend Servers do not access Account data directly, there is no need to make the SharedDomains file directory available ("mounted" or "mapped") to any Frontend Server.

Specify the addresses of all Backend Servers using the Frontend Server Settings->General->Cluster WebAdmin page.

To add a Frontend Server to your Dynamic Cluster, stop it, and restart it with the `--ClusterFrontend` Command Line option (it can be added to the CommuniGate Pro startup script). The Server will poll all specified Backend Server IP Addresses until it finds the active Cluster Controller.

Use the WebAdmin interface to verify that the Frontend Server is running. Use the Domains page to check that all Shared Domains are visible.

When Frontend Servers try to open one of the Shared Domain accounts, the Controller directs them to one of the running Backend Servers, distributing the load between all available Backend Servers.

---

# Shared Settings

The Dynamic Cluster maintains a separate set of "Default settings" for Shared Domains.

These settings include:

- Default Domain Settings for all Shared Domains
- Default Account Settings and Default WebUser Preferences for all Accounts in Shared Domains
- Cluster-Wide Alerts - these alerts are sent to all Accounts in Shared Domains

When the Server Administrator uses the WebAdmin Interface to modify these settings, the WebAdmin pages display the links that allow the Administrator to switch between the Server-wide settings (that work for all non-Shared Domains), and Cluster-wide settings. The Cluster-wide settings are automatically updated on all Cluster Members, and they work for all Shared Domains.

The Cluster-wide settings also include:

- Cluster-wide [Network](#) Settings.
- Cluster-wide [Router](#) Table.
- Cluster-wide [Directory Integration](#) Settings.
- Cluster-wide [Rules](#).
- Cluster-wide [Protection](#) Settings.
- Default and Named Cluster-wide [WebSkins](#). These WebSkins are used as default Skins for WebSkins in Shared Domains.
- Cluster-wide [Real-Time](#) Applications.
- Cluster-wide [Lawful Interception](#) settings.

# Shared Processing

The Dynamic Cluster Single Service Image component provides server synchronization beyond Account, Domain, and other Settings.

Additional "shared processing" functionality includes:

- Cluster-wide [Chronos](#) processing.
- Distributed [Queue](#) processing.
- Distributed [Signal](#) processing.
- Distributed [RPOP](#) and [RSIP](#) task processing.
- Cluster-wide [SMTP queue release](#) (via ETRN or Wakeup E-mail).
- Cluster-wide Authentication token (*nonce*) synchronization.
- Cluster-wide [Directory Units](#).

# Withdrawing Servers from a Dynamic Cluster

Use the WebAdmin Interface to withdraw a Server from a Dynamic Cluster. Open the Cluster page in the Monitors realm, and click the Make Non-Ready button.

When a Frontend Server is in the Non-Ready state, all its UDP ports and all its TCP ports (except the HTTP Admin ports) are closed.
The load balancer delivering incoming connections to the Cluster Frontends should detect this and stop sending new connections and packets to this Frontend.

When a Backend Server is in the Non-Ready state, the Controller does not send any new sessions to this Server. Wait until all existing sessions end, and then shut down the Backend Server.

If this Backend Server is the currently active Controller, then making it Non-Ready causes the Controller to send all new sessions to the other Backend Servers. If there are no other Backend Servers in the Cluster, the Controller continues to serve all new sessions itself.

You can click the Make Ready button on the same page to re-enable the Server. If the Server is a Backend, the Controller starts to send new sessions to it.

You need to have the [Can Control Cluster](#) access right to make Cluster members Ready or Non-Ready

If a Backend Server fails, all Shared Domain Accounts that were open on that Server at the time of failure become unavailable. They become available again within 5-10 seconds, when the Cluster Controller detects the failure. A

Backend Server failure does not cause any data loss.

# Upgrading Servers in a Dynamic Cluster

The Dynamic Cluster is designed to support "rolling upgrades". To upgrade to a newer version of the CommuniGate Pro software, you should upgrade the Servers one-by-one: withdraw a server from the Cluster, upgrade the software, and add the server back to the Cluster. This procedure allows your site to operate non-stop during the upgrade.

Certain changes in CommuniGate Pro software can impose some restrictions on the "rolling upgrade" process. Always check the History section before you upgrade your Cluster, and see if any Cluster Upgrade restrictions are specified there.

# Cluster Storage

- Storage Systems and File Systems
- Single OS File Systems
- Network File System (NAS)
- Storage Area Network (SAN)
- Cluster File Systems

The CommuniGate Pro Dynamic Clusters require *Shared File Systems*, so backend Cluster members can work with the same data files at the same time.

The most popular and well-known implementation of a Shared File System is a file server, also called NAS (*network attached storage*).

This section provides a brief overview of Shared File System technologies, explains why SAN (SAN (*storage area network*) is **not** a Shared File System, and provides an introduction to Cluster File Systems that can be used to build Shared File Systems using SAN.

## Storage Systems and File Systems

The Storage Systems (such as disk devices) used today are "dumb" devices from the user and application point of view. Each system or a device has some number of *blocks* - fixed-size data segments, for example 1K (1024 bytes) in size. When the disk device is connected to a computer, it can process only very simple requests, such as:

- READBLOCK(12345) - read the block number 12345 and send the block data to the computer.
- WRITEBLOCK(765645) - receive the data from the computer and store them in the block number 765645.

Disks can be connected to computers using IDE, SCSI, or FDDI interfaces. These interfaces are used to send commands and data to the disks, and to retrieve the data and command completion codes from the disks.

Storage Systems themselves do not create any other structures, meaning that a disk device cannot create "files" or "file directories". The only thing these systems work with are blocks, and all they can do is read and write those blocks.

## Single OS File Systems

Every modern Operating System (OS) has a component called a *File System*. That component is part of the OS *kernel*

and it implements things like "files" and "file directories".

There are many different File Systems, and they use various methods and algorithms, but the same basic functions are present in most File Systems:

- The File System maintains some sort of FAT (File Allocation Table) - information that associates logical files with storage block numbers.

  For example, the FAT can specify that the "File1" file is stored in 5 disk blocks with numbers 123400,123405,123401,177777,123456 and the "File2" file is stored in 6 disk blocks with numbers 323400,323405,323401,377777,323456, 893456.

- The File System maintains a list of all unused storage blocks and it automatically *allocates* new blocks when the file grows in size, and returns blocks into the list of unused blocks when a file decreases in size or when a file is deleted.
- The File System processes application requests that need to read from or write to logical files. The File System converts these requests into one or several storage block read and write operations, using the information in the File Allocation Table.
- The File System maintains special files called "file directories" and stores the information about other files in these directories.
- The File System maintains the "file cache." When new information is written to a file, it stores it in the Storage System (on disks) and it also copies this information into the File System "cache buffers".
  When file information is read from storage, it passes it to the application program and also copies it into the "cache buffers"
  When the same (or some other) application needs to read the same portion of the cached file, the File System simply retrieves that information from its cache buffers instead of re-reading it from the Storage System.

The following figure illustrates how a File System works:
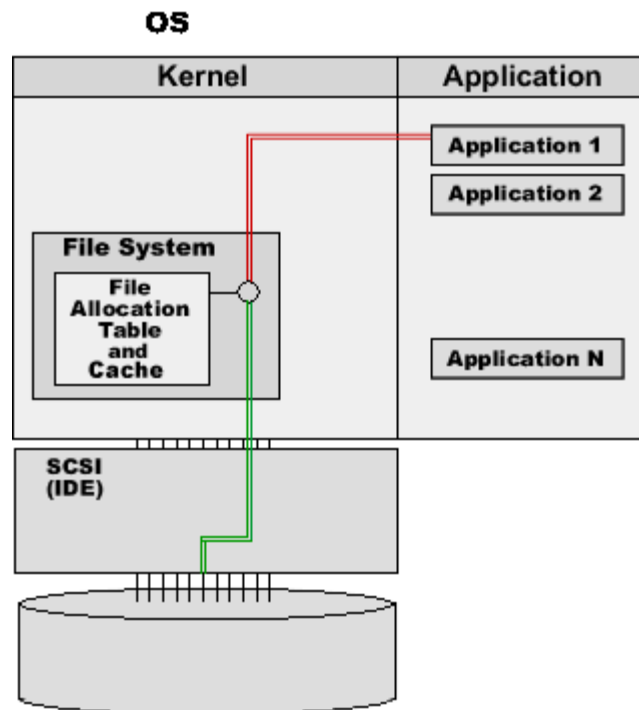


In this example, the File System serves requests from two applications.

**Application 1 asks the File System to read block number 5 from `File1`.**
The File System finds the information for File1 in the File Allocation Table, and detects that this file has 5 blocks allocated, and file block number 5 is stored in the block number 123456 on the disk.

The File System uses the disk interface (IDE, SCSI, or any other one) to send the READBLOCK(123456)

command to the disk.

The disk device sends the information from the specified block to the computer.

The File System places the read information into its cache buffers, and sends it to the application.

**Application 2 asks the File System to write block number 7 into `File2`.**

The File System finds the information for File2 in the File Allocation Table, and detects that this file has 6 blocks allocated. It checks the list of the unused disk blocks, and finds the unused block number 13477.

It removes the block number from the list of unused blocks and adds it as the 7th block to the File2 information in the File Allocation Table, so now File2 is 7 blocks in size.

The File System uses the disk interface (IDE, SCSI, or any other one) to send the WRITEBLOCK(13477) command to the disk, and sends the block data that the application program has composed.

The disk device writes the block data into the specified disk block, and confirms the operation.

The File System copies the block data information into its cache buffers.

If any application tries to read block 5 from File1 or block 7 from File2, the File System will retrieve the information from its cache buffers, and it will not perform any disk operation.

All applications running on this operating system use the same File System. The File System guarantees the data consistency. If the disk block 13477 is allocated to File2, it will not be allocated to any other file - until File2 is deleted or is decreased in size to less than 7 blocks.

# Network File System (NAS)

When server computers need to use the same data, a Network File System (also called NAS, or Network Attached Storage) can be used.

The Network File System is implemented using a *File Server* and a *network*. The File Server is a regular computer or specialized OS that has a regular File System and regular disk devices controlled with this File System.

The Network File System "stubs" running inside the OS kernel on "client" computers are "dummy" File Systems that retranslate application file requests to the File Server, using the network:
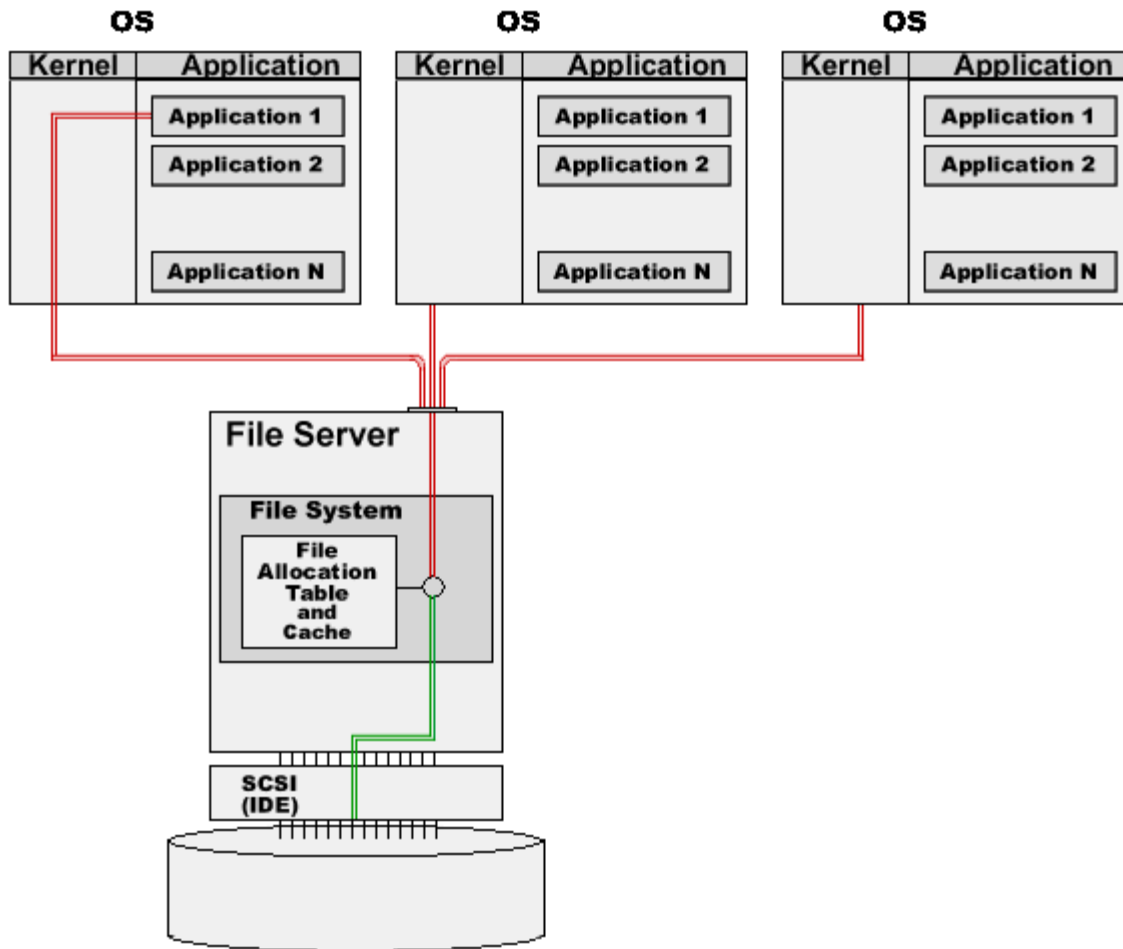
In this example, the File System on the File Server serves requests from several applications running on server "client" computers.

The only difference with the single OS is in the request delivery; instead of internal communication between an application and the File System running inside the OS kernel, the "stub" sends the requests via the network, receives the responses, and passes them to the application. All "real work" (File Allocation Table and cache maintenance) is done on the File Server computer.

Since only the File Server computer has direct access to the physical disk, all applications running on server systems use the same File System - the File System running on the File Server. That File System guarantees the data consistency. If the disk block 13477 is allocated to File2, it will not be allocated to any other file - until File2 is deleted or is decreased in size to less than 7 blocks.

# Storage Area Network

Storage Area Network is a special type of network that connects computers and disk devices; in the same way as SCSI cables connect disk devices to one computer.

Any computer connected to SAN can send disk commands to any disk device connected to the same SAN. On the physical level, SAN can be implemented using FDDI, Ethernet, or other types of networks.

Some disk drives or arrays have "dual-channel" SCSI controllers and can be connected to two computers using regular SCSI cables. Since both computers can send disk read/write commands to that shared disk, this configuration has the same functionality as a one-disk SAN.

SAN provides Shared Disks, but SAN itself does not provide a Shared File System. If you have several computers that have access to a Shared Disk (via SAN or dual-channel SCSI), and try to use that disk with a regular File System, the disk logical structure will be damaged very quickly.

There are two main problems with Shared Disks and regular File Systems:

**Disk Space Allocation inconsistency**

If computer X and computer Y both connected ("mounted") a shared disk, their File Systems loaded the File Allocation Tables into each computer's memory. Now, if some program running on computer X tried to write a new block to some file, the File System running on that computer will check its File Allocation Table and free blocks list, and it will allocate a new file block number 13477 to that file.

The File System running on that computer will modify its File Allocation Table, but it will have no effect on the File Allocation Tables loaded on other computers. If an application running on some other computer Y needs to expand a file, the File System running on that computer may allocate the same block 13477 to that other file, since it has no idea that this block has been already allocated by computer X.

**File Data inconsistency**

If a program running on computer X has read block 5 from some File1, that block is copied into the computer X File System Cache. If the same or another program running on computer X tries to read the same block 5 from the same file, the computer X File System will simply copy data from its cache.

A program running on some other computer Y can modify the information in the block 5 of File1. Since the File System running on computer X is not aware of this fact, it will continue to use its cache providing computer X applications with data that is no longer valid.

These problems make it impossible to use Shared Disks with regular File Systems as Shared File Systems. They can be used for fail-over systems or in any other configuration where only one computer is actually using the disk at any given time. The File System on computer Y starts to process the Shared Disk only when computer X has been shutdown, or stopped using the Shared Disk.

# Cluster File System

Cluster File Systems are software products designed to solve the problems outlined above. They allow you to build multi-computer systems with Shared Disks, solving the inconsistency problems.

Cluster File Systems are usually implemented as "wrapper" around some regular File System. Cluster File Systems use some kind of inter-server network to talk to each other and to synchronize their activities. That inter-server "interconnect" can be implemented using regular Ethernet networks, using the same SAN that connects computers and disks, or using special fast, low-latency "cluster interconnect" devices.
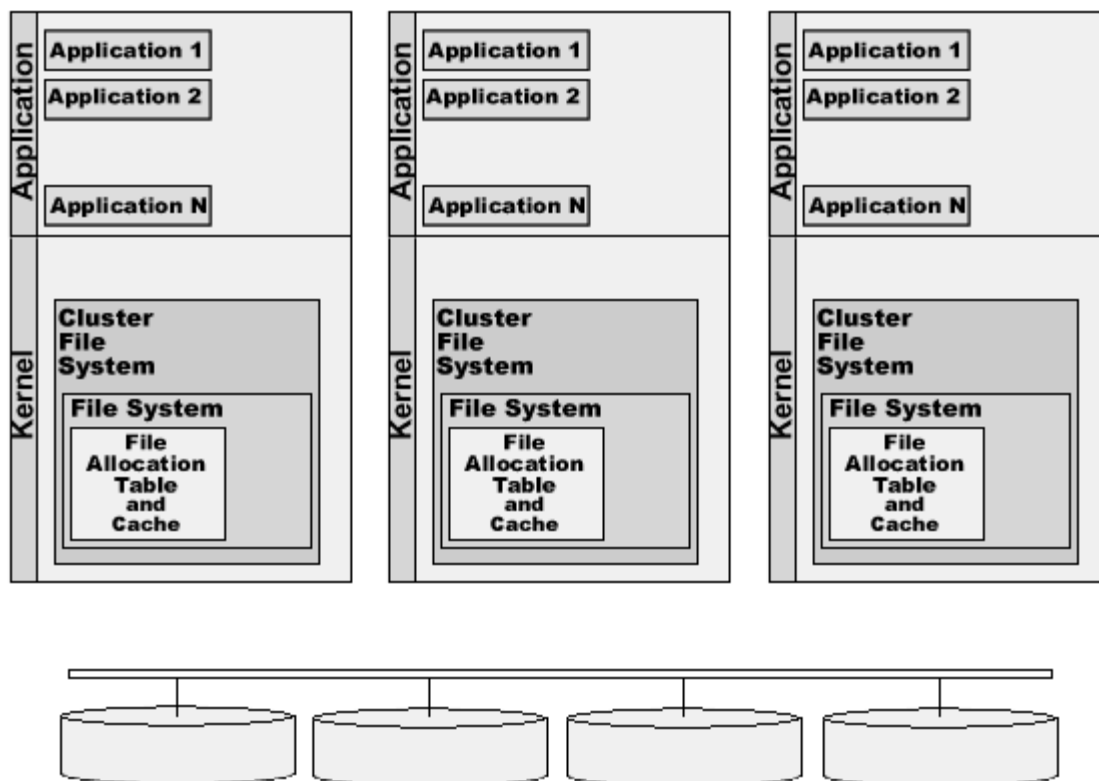
In this example, the Cluster File System is installed on several computers and serves requests from applications running on these computers.

**Application 1 running on the first computer asks the Cluster File System to read block number 5 from `File1`.**
The Cluster File system passes the request to the regular File System serving the Shared Disk, and the data block is read in the same way it is read on a single-server system.

**Application 2 running on a different system asks the Cluster File System to write block number 7 into `File2`.**
The Cluster file system uses the inter-server network to notify the Cluster File Systems on other computers that this block is being modified. The Cluster File Systems remove the old, obsolete copy of the block data from their caches.

The Cluster File System passes the request to the regular File System. It finds the information for File2 in the File Allocation Table, and detects that this file has 6 blocks allocated. It checks the list of unused disk blocks, and finds unused block number 13477. It removes the block number from the list of unused blocks and adds it as the 7th block to the File2 information in the File Allocation Table, so now File2 is 7 blocks in size.

The Cluster File System uses the inter-server network to notify the Cluster File Systems on other computers about the File Allocation Table modification. The Cluster File Systems on those computers update their File Allocation Tables to keep them in sync.

The File System uses the disk interface to send the WRITEBLOCK(13477) command to the Shared Disk, and sends the block data that the application program has composed.

The disk device writes the block data into the specified disk block, and confirms the operation.

The Cluster File System solves the inconsistency problems and allows several computers to use Shared Disk(s) as Shared File System.

Cluster File System products are available for several Operating Systems:

| Cluster File System | Operating System |
| --- | --- |
| Tru64 Cluster 5.x | HP Tru64 |
| VERITAS Cluster File System | Sun Solaris, HP/UX |
| Sun Cluster 3.0 | Sun Solaris |
| Generalized Parallel File System (GPFS) | IBM AIX, Linux |
| DataPlow | Linux, Solaris, Windows, IRIX |
| PolyServe | Linux |
| GFS | Linux |
| NonStop Cluster | Unixware |

# CommuniGate Pro

# E-Mail Transfer in Clusters

- **SMTP Relaying**
- **Local Delivery**
- **Backend Queues**
- **Remote Queue Processing**

This section explains how E-mail Message Transfer operations work in the CommuniGate Pro Cluster environment.

# SMTP Relaying

Incoming SMTP connections are received with a TCP Load Balancer and sent to Cluster Frontends. A Frontend Server receives a message in the same way as it does in the single-server mode, but it may contact the Backend Servers (via CLI) when it needs to:

- authenticate the sender
- verify the sender and recipient addresses
- check the recipient status
- retrieve the sender limits

Received messages are enqueued. If a message is directed to an external address, it can be relayed by the same Frontend:

LOAD BALANCER

FRONTEND SERVER **X**     FRONTEND SERVER **Y**

BACKEND SERVER **A**      BACKEND SERVER **B**

domain dom3.com
domain dom2.com
**domain dom1.com**
Account
Account
Account
**Account**

Other Mailboxes | **Inbox**
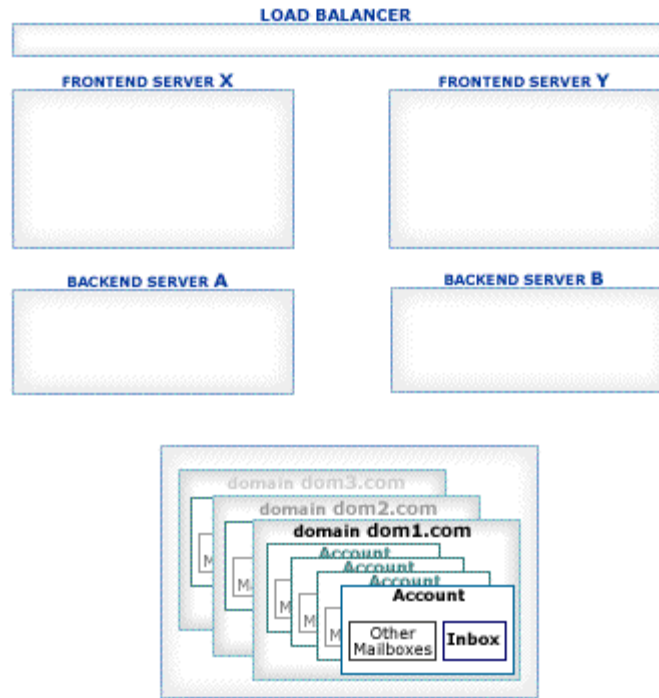
# Local Delivery

A message directed to a local recipient can be enqueued on a "wrong" Server, i.e. on a Server that cannot open the target Account and deliver the message to that Account.

This situation can happen when a message is enqueued on a Frontend Server (Frontend Servers cannot directly open any Account in Shared Domains), or a message is enqueued on a Backend Server that either does not host the target Account (in a Static Cluster), or it cannot open it, because the Account is opened by some other Backend Server (in a Dynamic Cluster).

To solve the problem, the Local Delivery module uses a Delivery channel connection to the proper Backend Server and passes the message there. The receiving Backend immediately opens the target Account, applies its Account-Level Rule and stores the transferred message. This Backend does not enqueue the message.

If there is a temporary problem or a delivery failure, the receiving Backend Server reports the error back, and the message is either delayed in Queue or it is removed from the Queue (with error report messages generated).

LOAD BALANCER

FRONTEND SERVER **X**    FRONTEND SERVER **Y**

BACKEND SERVER **A**    BACKEND SERVER **B**

domain dom3.com
domain dom2.com
**domain dom1.com**
Account
Account
Account
**Account**

Other Mailboxes | **Inbox**
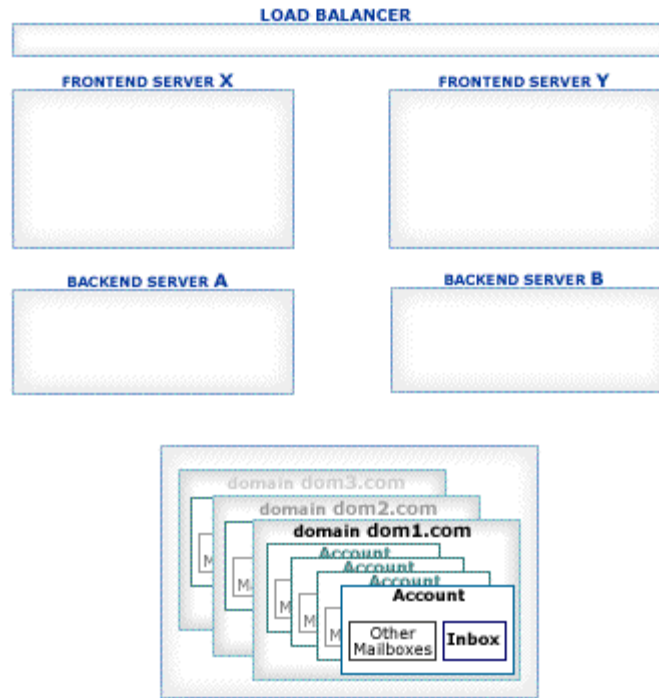
# Backend Queues

WebUser Interface, XIMSS, MAPI sessions, Rules, and other modules and components can generated E-mail messages on Backend Servers.

Backend Server often do not have direct access to the Internet, in this case they cannot deliver generated messages to remote systems. To solve this problem, the Backend Servers can be configured to relay all messages to the Frontend Servers first, using the * symbol as the SMTP Forwarding Server name.

In this case the message is submitted to the Backend Queue, where it is processed using the Server-wide and Cluster-wide Rules, and if it is not directed to a local recipient, it is directed to the SMTP module which sends it to one of the Frontend Servers:

BROWSER

LOAD BALANCER

FRONTEND SERVER X　　　FRONTEND SERVER Y

BACKEND SERVER A　　　BACKEND SERVER B

Session

domain dom3.com
domain dom2.com
domain dom1.com
Account
Account
Account
Account
Other Mailboxes　Inbox
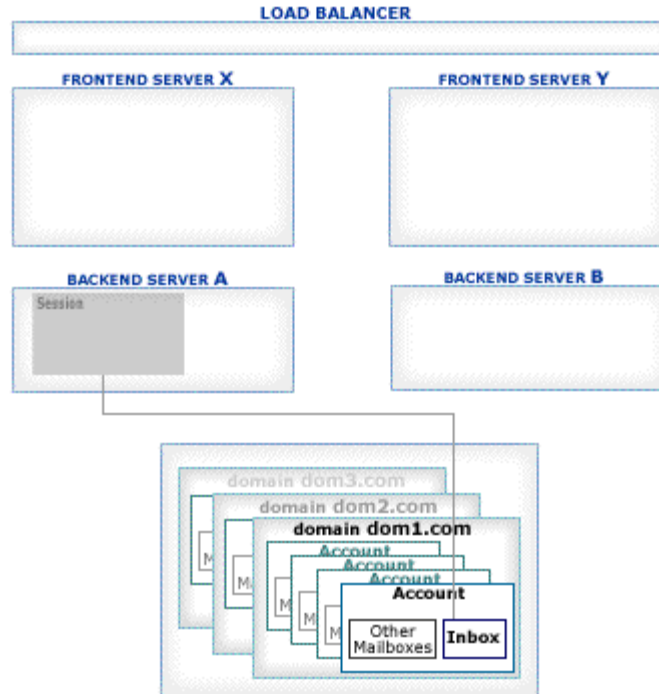
In this setup, each messages generated on a Backend Server is processed twice. If Cluster Rules invoke content management Plugins, double-processing can create a substantial overhead.

To avoid these problems and the inevitable overhead, the Remote Queue Processing method should be used.

# Remote Queue Processing

Most of the Queue processing takes place on the Frontend Servers. Frontend Servers accept incoming SMTP E-mail Messages and either relay them to remote locations or deliver them to local Accounts via Backends, using the special inter-cluster protocol, without placing messages into the Backend Server Queue.

A certain amount of E-mail Messages can be generated directly on the Backend Servers.

They include messages:

- composed with WebUser, XIMSS, AirSync, CalDAV sessions;
- submitted using the MAPI connector and XTND XMIT POP3 mechanism;
- created with Account-level Rules;
- retrieved using the RPOP module;
- submitted using the PIPE module.

You may want to avoid processing Message Queues on Backends for various reasons, including:

- lack of Internet connectivity for Backends;
- requirement to use anti-spam and/or anti-virus Plugins installed on Frontends only;
- shortage of processing power and/or disk space on Backend Servers.

You may also want to process Message Queues on some Frontends only.

To specify Queue processing options, open the Settings WebAdmin realm and select the Cluster link on the General page. Find the Queue Processing panel:

| Queue Processing | |
|---|---|
| Submit Messages:  Auto | Remote Submit Log:   Major & Failures |

Submit Messages

This setting specifies how the composed or received E-mail messages are submitted to the [Enqueuer component](#) for delivery.

Locally

messages are submitted to the Enqueuer component on the same Server (this is the "regular", single-server processing mode).

Locally for Others

messages are submitted to the Enqueuer component on the same Server.
The Dynamic Cluster Controller is informed that this Server can accept (enqueue) E-mail messages composed or received with the other Cluster members.
The Dynamic Cluster Controller collects and distributes information about all active Cluster members that have this option selected.

Remotely

messages are transferred to some Cluster member that has this setting set to Locally for Others. The temporary message file content (the message envelope and the message itself) is sent to that other Cluster member via a special protocol that uses the SMTP port. If the remote message submission does not work (the Server failed to connect to the Cluster member(s) or the message file transfer fails), the message is submitted to the Server own Queue to ensure that no message is lost:
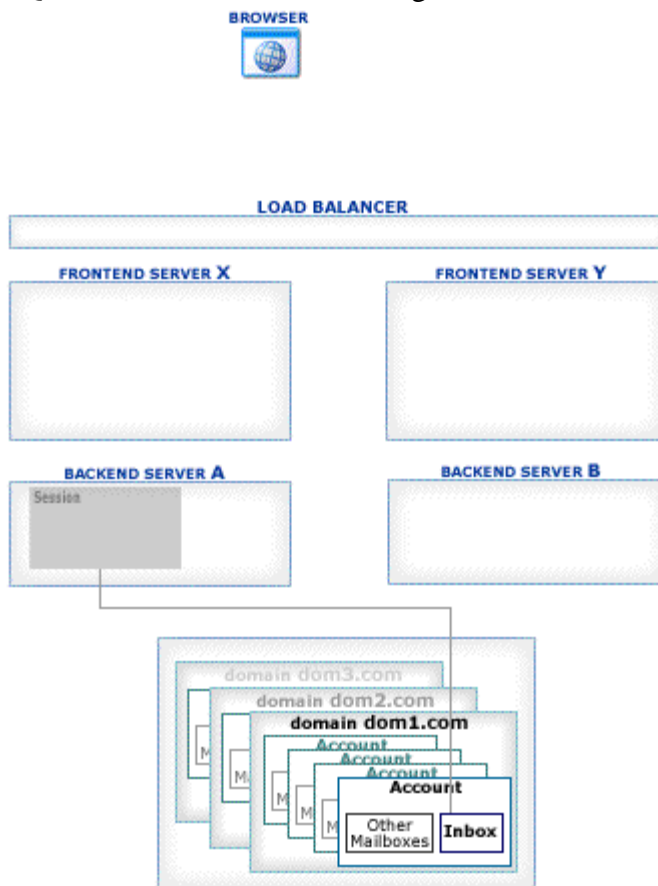


Auto

- if this Server is not a Dynamic Cluster member, same as `Locally`
- if this Server is a Dynamic Cluster frontend, same as `Locally for Others`
- if this Server is a Dynamic Cluster backend, same as `Remotely` if there are other Dynamic Cluster members configured as `Locally for Others`, if there are none - same as `Locally`

`Remote Submit Log`

Use this setting to specify what kind of information is stored in the Server Log when a message is being submitted remotely (to a different Server).

These records are marked with the `SUBMIT` tag.

---

Clusters Static Dynamic Storage E-Mail **Real-Time** Access Balancer

# Real-Time Processing in Clusters

- **Real-Time Tasks**
- **XIMSS Call Legs**
- **Signals**
- **Configuring Call Leg and Signal Processing**
- **SIP**
  - **Single-IP NAT Load Balancer**
  - **Multi-IP NAT Load Balancer**
  - **DSR Load Balancer**
- **RTP Media**
  - **Single-IP Method**
  - **Multi-IP No-NAT Load Balancer**
  - **Multi-IP NAT Method**
- **Sample Configurations**

This section explains how Real-Time operations work in the CommuniGate Pro Cluster environment:

- how Real-Time Tasks (such as Real-Time Applications and XIMSS Call Legs) communicate with each other, with XIMSS sessions, and with Real-Time Signals.
- how Real-Time Signals are created and access the Account data.
- how SIP transactions are handled.
- how RTP Media streams are processed.

# Real-Time Tasks

The CommuniGate Pro Real-Time Tasks communicate by sending *events* to *task handlers*. A *task handler* is a reference object describing a Real-Time Task, a Session, or a Real-Time Signal. In a CommuniGate Pro Cluster environment, the Task handler contains the address of the Cluster Member server the referenced Real-Time Task, Session, or Signal is running on.

When an event should be delivered to a different cluster member, it is delivered using the inter-Cluster CLI/API. The event recipient can reply, using the sender *task handler*, and again the inter-Cluster CLI/API will be used to deliver the reply event.

Real-Time Application Tasks usually employ Media channels. To be able to exchange media with external entities, Real-Time Tasks should run only on those Cluster members that have direct access to the Internet.

# XIMSS Call Legs

When a XIMSS session initiates a call, it creates a Call Leg object. These Call Leg objects manage XIMSS user Media channels and they should be able to exchange media with external entities, so they should run only on those Cluster members that have direct access to the Internet.

When a Real-Time Signal component directs an incoming call to a XIMSS session, it creates a Call Leg object on the same Cluster member processing this incoming call Signal request. This Call Leg object is then "attached" to the XIMSS session (which is running on some backend Server and this can be running on a different Cluster member).

When an XIMSS session and its Call Leg are running on different Cluster members, they communicate via special events, which are delivered using the inter-Cluster CLI/API.

# Signals

Real-Time Signal processing results in DNS Resolver , SIP, and XMPP requests.
When a Cluster is configured so that only the frontend servers can access the Internet, Real-Time Signal processing should take place on those frontend servers only.

Even if the Real-Time applications and Call Legs are configured to run on frontend servers only, Real-Time Signals can be generated on other cluster members, too: XIMSS and XMPP sessions, Automated Rules, and other components can send Instant Messages, Event packages generate notification Signals, etc.

When a Real-Time Signal is running on a frontend server, it uses inter-Cluster CLI/API to retrieve Account data (such as SIP registration), or to perform requested actions (to deliver SUBSCRIBE or XMPP IQ request, or to initiate a call).

# Configuring Call Leg and Signal Processing

To configure the Call Leg and Signal creation mode, open the General page in the Settings WebAdmin realm and click the Cluster link:

| Real-Time | |
|---|---|
| Call Legs Processing: | Auto |
| Signal Processing: | Auto |

Call Legs Processing
This setting specifies how the Real-Time Tasks and Call Leg objects should be created with this Cluster member.

Locally
when there is a request to create a Real-Time Task or a Call Leg object, it is created on the same Server (this is the "regular", single-server processing mode).

Locally for Others
Real-Time Task and Call Leg objects are created on the same Server.
The Dynamic Cluster Controller is informed that this Server can create Real-Time Task and Call Leg objects for other Cluster members.
The Dynamic Cluster Controller collects and distributes information about all active Cluster members that have this option selected.

Remotely
when there is a request to create a Real-Time Task or a Call Leg object, a request is relayed to some Cluster member that has this setting set to Locally for Others.

Auto
same as:

Locally
if this Server is not a Dynamic Cluster member.

```
Locally for Others
```
if this Server is a Dynamic Cluster frontend.

```
Remotely
```
if this Server is a Dynamic Cluster backend.

```
Signal Processing
```
This setting specifies how the Signal objects should be created with this Cluster member. Values for this setting have the same meaning as for the Call Legs Processing setting.

---

# SIP

The CommuniGate Pro SIP Farm® feature allows several Cluster members to process SIP request packets randomly distributed to them by a Load Balancer.

Configure the Load Balancer to distribute incoming SIP UDP packets (port 5060 by default) to the SIP ports of the selected SIP Farm Cluster members.
If your Cluster has Frontend Servers, then all or some of the Frontend Servers should be used as SIP Farm members.

To configure the SIP Farm Members, open the General page in the Settings WebAdmin realm and click the Cluster link:

| Real-Time |
| --- |
| SIP Farm:    Auto |

```
SIP Farm
```
This setting specifies how the SIP requests should be processed by this Cluster member.

```
Member
```
If this option is selected, this Cluster member is a member of a SIP Farm. It processes new requests locally or it redirects them to other SIP Farm members based on the SIP Farm algorithms.

```
Disabled
```
If this option is selected, this Cluster member is not a member of a SIP Farm; it will process incoming SIP requests locally.

```
Relay
```
If this option is selected, this Cluster member is not a member of a SIP Farm; but when it needs to send a SIP request, it will relay it via the currently available SIP Farm members.
Select this option for Backend Servers that do not have direct access to the Internet and thus cannot send SIP requests directly.

```
Auto
```

- if this Server is not a Dynamic Cluster member, same as `Disabled`
- if this Server is a Dynamic Cluster frontend, same as `Member`
- if this Server is a Dynamic Cluster backend, same as `Relay` if there are other Dynamic Cluster members configured as `Member`, if there are none - same as `Disabled`

**Note:** a SIP request can explicitly address some Cluster member (most in-dialog requests do). These requests are always redirected to the specified Cluster member and processed on that member, regardless of the SIP Farm settings.

The CommuniGate Pro Cluster maintains the information about all its Servers with the SIP Farm setting set to Member. Incoming UDP packets and TCP connections are distributed to those Servers using regular simple Load Balancers.

The receiving Server detects if the received packet must be processed on a certain Farm Server: it checks if the packet is a response or an ACK packet for an existing transaction or if the packet is directed to a Node created on a certain Server. In this case the packet is relayed to the proper Cluster member:
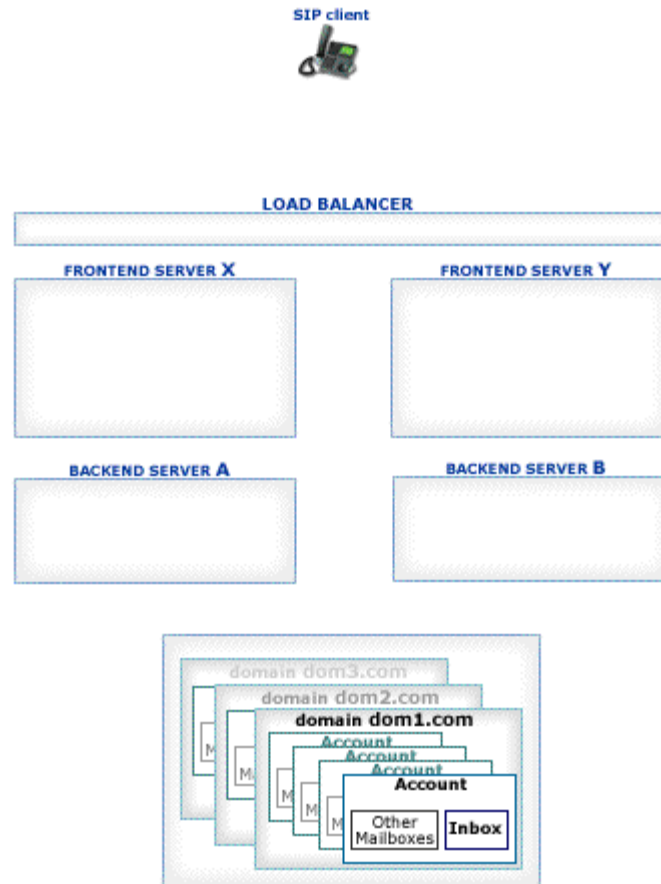


Packets not directed to a particular Cluster member are distributed to all currently available Farm Members based on the CommuniGate Pro SIP Farm algorithms.

To process a Signal, Cluster members may need to retrieve certain Account information (registration, preferences, etc.). If the Cluster member cannot open the Account (because the Member is a Frontend Server or because the Account is locked on a different Backend Server), it uses the inter-Cluster CLI/API to retrieve the required information from the proper Backend Server.

Several Load Balancer and network configurations can be used to implement a SIP Farm:

# Single-IP NAT Load Balancer

This method is used for small Cluster installations, when the frontend Servers do not have direct access to the Internet, and the Load Balancer performs Network Address Translation for frontend Servers.

First select the "virtual" IP address (VIP) - this is the only address your Cluster SIP users will "see":

- assign the VIP address to the Load Balancer
- select a "sip-service" DNS domain name (such as `sip.mysystem.com`), and create a DNS A- or AAAA- record for that name, pointing to the VIP address.
- create DNS SIP SRV records for all your Cluster Domains pointing to this "sip-service" name.
- open the Network pages in the Settings realm of the CommuniGate Pro WebAdmin Interface, and specify the VIP address as the Cluster-wide WAN IP address; **leave the Server-wide WAN IP Address field empty.**

The frontend servers have IP addresses F1, F2, F3, ...

Configure the Load Balancer to process incoming UDP packets received on its VIP address and port 5060:

- incoming packets should be redirected evenly to F1, F2, F3 frontend server addresses, to the same port 5060.
- the Load Balancer should not apply any SIP-specific logic to these packets; **if your Load Balancer has any SIP-specific options, make sure they are switched off.** Some Load Balancers use SIP-specific processing for port 5060 by default: consult with your Load Balancer manufacturer.
- incoming packets should not create any "session" in the Load Balancer, i.e. the Load Balancer should not keep any information about an incoming UDP packet after it has been redirected to some frontend server.

SIP-specific techniques implemented in some Load Balancers allow them to send all "related" requests to the same server. Usually these techniques are based on the request Call-ID field and thus fail very often. CommuniGate Pro SIP Farm technology ensures proper request handling if a request or response packet is received by any SIP Farm member. Thus, these SIP-specific Load Balancer techniques are not required with CommuniGate Pro.

Many Load Balancers create "session binding" for incoming UDP requests, in the same way they process incoming TCP connections - even if they do not implement any SIP-special techniques.
The Binding table for some Load Balancer port v (and the Load Balancer VIP address) contains IP address-port pairs:

`X:x <-> F1:f`
where X:x is the remote (sending) device IP address and port, and F1:f is the frontend Server IP address and port the incoming packet has been forwarded to.
When the remote device re-sends the request, this table record allows the Load Balancer to send the request to the same frontend Server (note that this is **not** needed with the CommuniGate Pro SIP Farm).

These Load Balancers usually create "session binding" for outgoing UDP requests, too: when a packet is sent from some frontend address/port F2:f to some remote address/port Y:y, a record is created in the Load Balancer Binding table:

`Y:y <-> F2:f`

When the remote device sends a response packet, this table record allows the Load Balancer to send the response to the "proper" frontend Server (note that this is **not** needed with the CommuniGate Pro SIP Farm).

CommuniGate Pro SIP Farm distributes SIP request packets by relaying them between the frontend Servers, according to the SIP Farm algorithms; the SIP Farm algorithms redirect the SIP response packets to the frontend Server that has sent the related SIP request.
These CommuniGate Pro SIP Farm features make the Load Balancer "session binding" table useless (when used for SIP UDP)

The Load Balancer "session binding" must be switched off (for SIP UDP), because it not only creates unnecessary overhead, but it usually corrupts the source address of the outgoing SIP packets:

When a Load Balancer receives a SIP request packet from X:x address, and relays it to the frontend Server F1:5060 address/port, the SIP Farm can relay this request to some other frontend Server (the F2:5060 address/port), where a SIP Server transaction will be created and the request will be processed.

SIP responses will be generated with this frontend Server, and the SIP response packets will be sent out to X:x from the F2:5060 address/port (via the Load Balancer).

If the Load Balancer does not do any "session binding", it should simply change the packet source address from F2:5060 to VIP:5060, and redirect it to X:x.

If the Load Balancer does implement UDP "session binding", it expects to see the response packets from the same F1:5060 address only; it will then redirect them to X:x after changing the response packet source address from F1:5060 to VIP:5060.

Packets from other servers (for which it does not have a "session binding") are processed as "outgoing packets", and a Load Balancer builds a new "session binding" for them (see above). In our case, when a Load Balancer sends a request from X:x to F1:5060, and gets a response from F2:5060, it would have to create the second "session binding":

```
X:x <-> F1:5060
X:x <-> F2:5060
```

These are conflicting "session binding" for most Load Balancers, and in order to solve the conflict the Load Balancer will use NAT techniques and change not only the source address of the outgoing packet, but its source port, too - so the response packet will be sent to X:x with the source address set not to VIP:5060, but to VIP:5061 (or any other source port the Load Balancer uses for NAT). Many SIP devices, and most SIP devices behind firewalls will not accept responses from the VIP:5061 address/port, if they have sent requests to VIP:5060 address/port.

It is very important to consult with your Load Balancer manufacturer to ensure that the Load Balancer does not use "session binding" for UDP port 5060 - to avoid the problem described above.

# Multi-IP NAT Load Balancer

In this configuration frontend Servers have direct access to the Internet (they have IP addresses directly "visible" from the Internet).

- The Load Balancer redirects incoming requests to these real F1, F2, F3... frontend Server addresses.
- The Load Balancer should be implemented as the a switch, so outgoing traffic from frontend Servers will pass via the Load Balancer.
- The Load Balancer should change the source IP of all outgoing SIP UDP packets coming from frontend Servers (from Fn:5060) to VIP:5060.

Load Balancers with UDP "session binding" will have the same problems as described above.

# DSR Load Balancer

DSR (Direct Server Response) is the preferred Load-Balancing method for larger installations.

To use the DSR method, create an "alias" for the loopback network interface on each Frontend Server. While the standard address for the loopback interface is 127.0.0.1, create an alias with the VIP address and the 255.255.255.255 network mask:

**Solaris**
```
ifconfig lo0:1 plumb
ifconfig lo0:1 VIP netmask 255.255.255.255 up
```
To make this configuration permanent, create the file /etc/hostname.lo0:1 with the VIP address in it.

**Linux**
```
ifconfig lo:0 VIP netmask 255.255.255.255 up
```
To make this configuration permanent, create the file /etc/sysconfig/network-scripts/ifcfg-lo:0:

```
DEVICE=lo
IPADDR=VIP
NETMASK=255.255.255.255
ONBOOT=yes
```

Make sure that the kernel is configured to avoid ARP advertising for this `lo` interface (so the `VIP` address is not linked to any Frontend server in arp-tables). Subject to the Linux kernel version, the following commands should be added to the `/etc/sysctl.conf` file:

```
# ARP: reply only if the target IP address is
# a local address configured on the incoming interface
net.ipv4.conf.all.arp_ignore = 1
#
# When an arp request is received on eth0, only respond
# if that address is configured on eth0.
net.ipv4.conf.eth0.arp_ignore = 1
#
# Enable configuration of arp_announce option
net.ipv4.conf.all.arp_announce = 2
# When making an ARP request sent through eth0, always use an address
# that is configured on eth0 as the source address of the ARP request.
net.ipv4.conf.eth0.arp_announce = 2
#
# Repeat for eth1, eth2 (if exist)
#net.ipv4.conf.eth1.arp_ignore = 1
#net.ipv4.conf.eth1.arp_announce = 2
#net.ipv4.conf.eth2.arp_ignore = 1
#net.ipv4.conf.eth2.arp_announce = 2
```

**FreeBSD**

To change the configuration permanently, add the following line to the `/etc/rc.conf` file:

```
ifconfig_lo0_alias0="inet VIP netmask 255.255.255.255"
```

*other OS*

consult with the OS vendor

- When this "alias" is created, the frontend Servers do not respond to the "arp" requests for the VIP address, and all packets directed to the VIP address will be routed to the Load Balancer.
- When the Load Balancer uses the "DSR" method, it does not change the target IP address (VIP) of the incoming packets. Instead the Load Balancer redirects incoming packets using the network-level (MAC) address of the selected frontend Server.
- Because the frontend Server has the VIP address configured on one of its interfaces (on the loopback interface), it accepts this packet as a local one, and passes it to the application listening on the specified TCP or UDP port.
- Because the frontend Server has the VIP address configured on one of its interfaces, response and other outgoing packets can be sent using the VIP address as the source address. If these packets come via the Load Balancer (they can bypass it), the Load Balancer should not modify them in any way.

**Note:** Because MAC addresses are used to redirect incoming packets, the Load Balancer and all frontend Servers must be connected to the same network segment; there should be no router between the Load Balancer and frontend Servers.

**Note:** when a network "alias" is created, open the General Info page in the CommuniGate Pro WebAdmin Settings realm, and click the Refresh button to let the Server detect the newly added IP address.

The DSR method is transparent for all TCP-based services (including SIP over TCP/TLS), no additional CommuniGate Pro Server configuration is required: when a TCP connection is accepted on a local VIP address, outgoing packets for that connection will always have the same VIP address as the source address.

To use the DSR method for SIP UDP, the CommuniGate Pro frontend Server configuration should be updated:

- use the WebAdmin Interface to open the Settings realm. Open the SIP receiving page in the Real-Time section

follow the UDP Listener link to open the Listener page
- by default, the SIP UDP Listener has one socket: it listens on "all addresses", on the port 5060.
- change this socket configuration by changing the "all addresses" value to the *VIP* value (the *VIP* address should be present in the selection menu).
- click the Update button
- create an additional socket to receive incoming packets on the port 5060, "all addresses", and click the Update button

Now, when you have 2 sockets - the first socket for *VIP*:5060, the second one for *all addresses*:5060, the frontend Server can use the first socket when it needs to send packets with VIP source address.
Repeat this configuration change for all frontend Servers.

---

# RTP Media

Each Media stream terminated in CommuniGate Pro (a stream relayed with a *media proxy* or a stream processed with a *media server channel*) is bound to a particular Cluster Member. The Load Balancer must ensure that all incoming Media packets are delivered to the proper Cluster Member.

## Single-IP Method

The "single-IP" method is useful for a small and medium-size installations.
The Cluster Members have internal addresses L1, L1, L3, etc.
The Load Balancer has an external address G0.
The Network Settings of each Cluster Member are modified, so the Media Ports used on each Member are different: ports 10000-19999 on the L1 Member, ports 20000-29999 on the L2 Member, ports 30000-39999 on the L3 Member, etc.
All packets coming to the G0 address to the standard ports (5060 for SIP) are distributed to the L1, L2, L3 addresses, to the same ports.
All packets coming to the G0 address to the media ports are distributed according to the port range:

- packets coming to the ports 10000-19999 are directed to the L1 address (without port number change)
- packets coming to the ports 20000-29999 are directed to the L2 address (without port number change)
- packets coming to the ports 30000-39999 are directed to the L3 address (without port number change)

The Server-wide WAN IP Address setting should be left empty on all Cluster Members.
The Cluster-wide WAN IP Address setting should specify the G0 address.

This method should not be used for large installations (unless there is little or no media termination): it allows you to allocate only 64000 ports for all Cluster media streams (each AVP stream takes 2 ports, so the total number of audio streams is limited to 32000, and if video is used (together with audio), such a Cluster cannot support more than 16,000 concurrent A/V sessions.

## Multi-IP No-NAT Load Balancer

The "multi-IP" method is useful for large installations. Each frontend has its own IP address, and when a Media Channel or a Media Proxy is created on that frontend Server, this unique IP address is used for direct communication between the Server and the client device or remote server.

The Network Settings of each Cluster Member can specify the same Media Port ranges, and the number of concurrent RTP streams is not limited by 64000 ports.

In the simplest case, all frontend Servers have "real" IP Addresses, i.e. they are directly connected to the Internet.

If the Load Balancer uses a DSR method (see above), then it should not care about the packets originating on the frontend Servers from non-VIP addresses: these packets either bypass the Load Balancer, or it should deliver them without any modification.

If the Load Balancer uses a "normal" method, it should be instructed to process "load balanced ports" only, while packets to and from "other ports" (such as the ports in the Media Ports range) should be redirected without any modification.

## Multi-IP NAT Method

You can use the Multi-IP method even if your frontend Servers do not have "real" IP Addresses, but they use "LAN"-type addresses `L1`, `L1`, `L3`, etc.

Configure the Load Balancer to host real IP Addresses G1, G2, G3,... - in addition to the VIP IP Address used to access CommuniGate Pro services.

Configure the Load Balancer to "map" its external IP address G1 to the frontend Server address L1, so all packets coming to the IP Address G1, port g (G1:g) are redirected to the frontend Server address L1, same port g (L1:g). The Load Balancer may change the packet target address to L1, or it may leave it as is (G1); When the Load Balancer receives a packet from the L1 address, port l (L1:l), and this port is not a port involved in a load balancing operations (an SMTP, POP, IMAP, SIP, etc.), the Load Balancer should redirect the packet outside, replacing its source address from L1 to G1: L1:l->G1:l.

Configure the Load Balancer in the same way to "map" its external IP addresses G2, G3, ... to the other frontend Server IP addresses L2, L3...

Configure the CommuniGate Pro frontend Servers, using the WebAdmin Settings realm. Open the Network pages, and specify the "mapped" IP addresses as Server-wide WAN IP Addresses: G1 for the frontend Server with L1 IP address, G2 for the frontend Server with L2 IP address, etc.

Introduction

Installation

SysAdmin

Network

Objects

Storage

E-mail

Real-Time

Access

Services

Directory

Clusters

Applications

Miscellaneous

Licensing

WebMail

Pronto!

PBX

| Clusters | Static | Dynamic | Storage | E-Mail | Real-Time | Access | Balancer |

# Account Access in Clusters

- **POP, IMAP, MAPI, ACAP, XMPP Interfaces**
- **File Access (FTP, TFTP, HTTP) Interfaces**
- **Service (RADIUS, LDAP, PWD) Interfaces**
- **WebUser Interface**

This section explains how Account Access operations work in the CommuniGate Pro Cluster environment.

The CommuniGate Pro Cluster architecture allows a load balancer to direct any connection to any working Server, eliminating a need for complex and unreliable "high-level" load balancer solutions. Inexpensive Layer 4 Switches can be used to handle the traffic.

# POP, IMAP, MAPI, ACAP, XMPP Interfaces

POP, IMAP, MAPI, ACAP, XMPP sessions are created on the Backend server that succeeds to open the target Account.

These protocols work over the TCP network protocol. When a TCP connection is established to the Server that cannot open the target Account (to a Frontend Server, or to a "wrong" Backend server), the Server builds a protocol proxy and connects the client application with the proper Server.

If the connection is encrypted (using SSL/TLS), request decryption and response encryption take place on the frontend server:

---
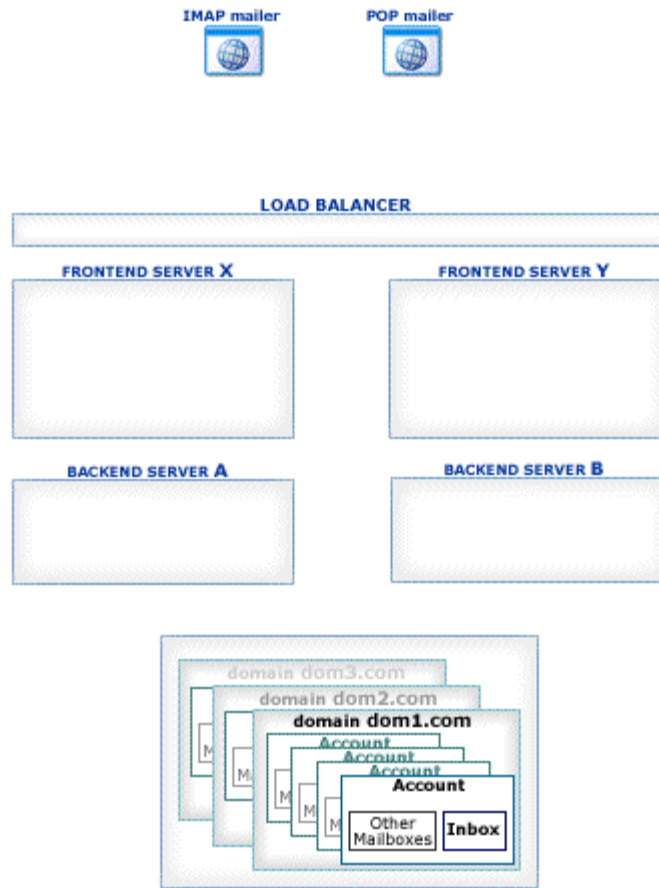
# File Access (FTP, TFTP, HTTP) Interfaces

When an FTP connection is established, or when a TFTP or HTTP request is received, the protocol session is created on the same Server.

Inter-cluster CLI is used to authenticate the user and/or to access the Account data if the user Account cannot be opened on the same Server.

---

# Service (RADIUS, LDAP, PWD) Interfaces

When a Service request is received, it is processed on the same Server.

Inter-cluster CLI is used to authenticate the user and/or to access the Account data if the user Account cannot be opened on the same Server.

---

# WebUser Interface

WebUser Interface Sessions are created on the Backend server that succeeds to open the target Account.

User browsers send HTTP requests to the Frontend Servers via Load Balancer(s). If the session request hits the "wrong" Server (i.e. the server that does not host the target session), the request is proxied to the correct Server.

If the HTTP connection is encrypted (using SSL/TLS), request decryption and response encryption take place on the

frontend server:
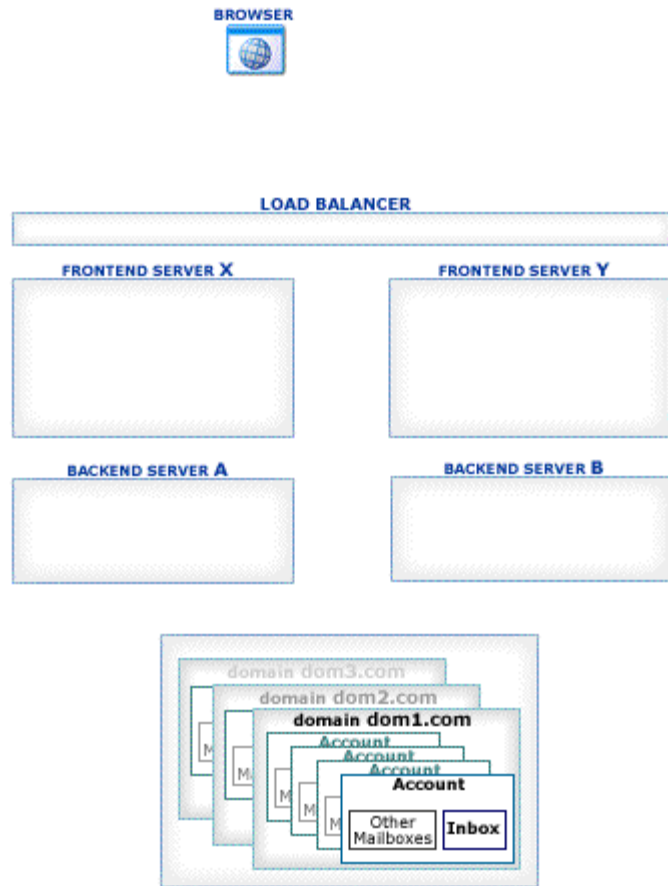
| Clusters | Static | Dynamic | Storage | E-Mail | Real-Time | Access | Balancer |
|---|---|---|---|---|---|---|---|

# Cluster Load Balancers

- **DSR (Direct Server Response) or DR (Direct Routing)**
- **Pinging**
- **Sample Balancer Configurations**
- **Outgoing TCP Connections**
- **Software Load Balancer**
    - **Linux IPVS**

The CommuniGate Pro Cluster architecture allows a load balancer to direct any connection to any working Server, eliminating a requirement for complex and unreliable "high-level" load balancer solutions. Inexpensive Layer 4 Switches can be used to handle the traffic.

Additionally, CommuniGate Pro Dynamic Cluster can manage software load balancers, such as the Linux "ipvs" kernel module: the CommuniGate Pro Cluster collects the information about working Servers belonging to one or several "balancer groups", learns which Servers can be used as Load Balancers, selects one Load Balancer for each group, informs it about all active Servers in that group, and assigns the Load Balancer duties to a different server, if the selected one goes down.

# DSR (Direct Server Response) or DR (Direct Routing)

The DSR/DR is the preferred Load-Balancing method for larger installations. When this method is used, each Server is configured to have the VIP (Virtual IP) shared addresses as its local IP addresses. This allows each Server to receive all packets directed to the VIP addresses, and to send responses directly to the clients using the VIP as the "source" address.
The servers should not respond to the arp requests for these VIP addresses. Instead the load balancer responds to these requests, and thus all incoming packets directed to the VIP addresses are delivered to the load balancer, which redirects them to Servers. When redirecting these incoming packets, the load balancer sends them directly to the Server MAC address, without changing the packet destination address, that remains the VIP address.

**Note:** Because MAC addresses are used to redirect incoming packets, the Load Balancer and all balanced Servers (usually - CommuniGate Pro Cluster frontends) must be connected to the same network segment; there should be no router between the Load Balancer and those Servers.

To use the DSR method, create an "alias" for the loopback network interface on each Frontend Server. While the standard address for the loopback interface is 127.0.0.1, create an alias with the VIP address and the 255.255.255.255 network mask:

**Solaris**

```
ifconfig lo0:1 plumb
ifconfig lo0:1 VIP netmask 255.255.255.255 up
```

To make this configuration permanent, create the file `/etc/hostname.lo0:1` with the VIP address in it.

### FreeBSD

To change the configuration permanently, add the following line to the `/etc/rc.conf` file:

```
ifconfig_lo0_alias0="inet VIP netmask 255.255.255.255"
```

### Linux

```
ifconfig lo:0 VIP netmask 255.255.255.255 up
```

or

```
ip address add VIP/32 dev lo
```

To make this configuration permanent, create the file `/etc/sysconfig/network-scripts/ifcfg-lo:0`:

```
DEVICE=lo
IPADDR=VIP
NETMASK=255.255.255.255
ONBOOT=yes
```

Make sure that the kernel is configured to avoid ARP advertising for this `lo` interface (so the `VIP` address is not linked to any Frontend server in arp-tables). Subject to the Linux kernel version, the following commands should be added to the `/etc/sysctl.conf` file:

```
# ARP: reply only if the target IP address is
# a local address configured on the incoming interface
net.ipv4.conf.all.arp_ignore = 1
#
# When an arp request is received on eth0, only respond
# if that address is configured on eth0.
net.ipv4.conf.eth0.arp_ignore = 1
#
# Enable configuration of arp_announce option
net.ipv4.conf.all.arp_announce = 2
# When making an ARP request sent through eth0, always use an address
# that is configured on eth0 as the source address of the ARP request.
net.ipv4.conf.eth0.arp_announce = 2
#
# Repeat for eth1, eth2 (if exist)
#net.ipv4.conf.eth1.arp_ignore = 1
#net.ipv4.conf.eth1.arp_announce = 2
#net.ipv4.conf.eth2.arp_ignore = 1
#net.ipv4.conf.eth2.arp_announce = 2
```

If you plan to have many VIPs, or if you plan to use CommuniGate Pro Load Balancing with the Linux built-in ipvs load balancer, do not create `/etc/sysconfig/network-scripts/ifcfg-lo:n` files.
Create the `/etc/sysconfig/vipaddrs` configuration file instead, and put all VIP addresses into it, as addresses, or subnetworks, one address per line. For example:

```
# single addresses
72.20.112.45
72.20.112.46
# a subnetwork
72.20.112.48/29
```

**Note:** line starting with the `#` symbol are ignored. They can be used as comments.

**Note:** subnetwork masks must be 24 bits or longer.

Create the following configuration scripts:

```
/etc/sysconfig/network-scripts/ifvip-utils
```

```bash
#!/bin/bash
#
# /etc/sysconfig/network-scripts/ifvip-utils
#
VIPADDRFILE="/etc/sysconfig/vipaddrs"

VIPLIST=""              # list of VIP masks: xxx.yy.zz.tt/mm where mm should be >= 24

for xVIP in `cat $VIPADDRFILE | grep -v '^#'`; do
  if [[ $xVIP != */* ]]; then xVIP=$xVIP/32; fi
  if (( ${xVIP##*/} < 24)); then
    echo "Incorrect mask: $xVIP" >2 ; exit 1;
  fi
  VIPLIST="$VIPLIST$xVIP "
done

CURRENT=`ip address show dev lo | egrep '^ +inet [0-9]+\.[0-9]+\.[0-9]+\.[0-9]+\/32 .*$' |
sed -r 's/ +inet ([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+).*/\1/' `

function contains() {
  local x;
  for x in $1; do
    if [[ $x == $2 ]]; then return 0; fi
  done
  return 1
}
```

/etc/sysconfig/network-scripts/ifup-lo

```bash
#!/bin/bash
#
# /etc/sysconfig/network-scripts/ifup-lo
#
/etc/sysconfig/network-scripts/ifup-eth ${1} ${2}
#
# Bring up all addresses listed in the VIPADDRFILE file, as lo aliases
#
. /etc/sysconfig/network-scripts/ifvip-utils

for xVIP in $VIPLIST; do
  xIP=${xVIP%/*}         # xx.xx.xx.yy/mm -> xx.xx.xx.yy
  xIP0=${xIP%.*}         # xx.xx.xx.yy/mm -> xx.xx.xx
  xIP1=${xIP##*.}        # xx.xx.xx.yy/mm -> yy
  xMask=$(( 2 ** (32 - ${xVIP##*/}) ))
  for (( index=0; index<$xMask; index++ )); do
    thisIP=$xIP0.$((xIP1 + index))
    if ! $(contains "$CURRENT" "$thisIP"); then
      ip address add $thisIP/32 dev lo
    fi
  done
done
```

/etc/sysconfig/network-scripts/ifdown-lo

```bash
#!/bin/bash
#
# /etc/sysconfig/network-scripts/ifdown-lo
#
# Bring down all addresses listed in the VIPADDRFILE file
#
. /etc/sysconfig/network-scripts/ifvip-utils

for xVIP in $VIPLIST; do
  xIP=${xVIP%/*}         # xx.xx.xx.yy/mm -> xx.xx.xx.yy
  xIP0=${xIP%.*}         # xx.xx.xx.yy/mm -> xx.xx.xx
  xIP1=${xIP##*.}        # xx.xx.xx.yy/mm -> yy
  xMask=$(( 2 ** (32 - ${xVIP##*/}) ))
  for (( index=0; index<$xMask; index++ )); do
    thisIP=$xIP0.$((xIP1 + index))
    if $(contains "$CURRENT" "$thisIP"); then
      ip address delete $thisIP/32 dev lo
    fi
```

```
      done
   done

/etc/sysconfig/network-scripts/ifdown-eth ${1} ${2}
```

*other OS*
> consult with the OS vendor

**Note:** when a network "alias" is created, open the General Info page in the CommuniGate Pro WebAdmin Settings realm, and click the Refresh button to let the Server detect the newly added IP address.

The DSR method is transparent for all TCP-based services (including SIP over TCP/TLS), no additional CommuniGate Pro Server configuration is required: when a TCP connection is accepted on a local VIP address, outgoing packets for that connection will always have the same VIP address as the source address.

To use the DSR method for SIP UDP, the CommuniGate Pro frontend Server configuration should be updated:

- use the WebAdmin Interface to open the Settings realm. Open the SIP receiving page in the Real-Time section
- follow the UDP Listener link to open the Listener page
- by default, the SIP UDP Listener has one socket: it listens on "all addresses", on the port 5060.
- change this socket configuration by changing the "all addresses" value to the *VIP* value (the *VIP* address should be present in the selection menu).
- click the Update button
- create an additional socket to receive incoming packets on the port 5060, "all addresses", and click the Update button

Now, when you have 2 sockets - the first socket for *VIP*:5060, the second one for *all addresses*:5060, the frontend Server can use the first socket when it needs to send packets with VIP source address.
Repeat this configuration change for all "balanced" Servers.

# Pinging

Load Balancers usually send some requests to servers in their "balanced pools". Lack of response tells the Load Balancer to remove the server from the pool, and to distribute incoming requests to remaining servers in that pool.

With SIP Farming switched on, the Load Balancer own requests can be relayed to other servers in the SIP Farm, and responses will come from those servers. This may cause the Load Balancer to decide that the server it has sent the request to is down, and to exclude the server from the pool.
To avoid this problem, use the following SIP requests for Load Balancer "pinging":

```
OPTION sip:aaa.bbb.ccc.ddd:5060 SIP/2.0
Route: <sip:aaa.bbb.ccc.ddd:5060;lr>
other SIP packet fields
```

where *aaa.bbb.ccc.ddd* is the IP address of the CommuniGate Pro Server being tested.

These packets are processed with the *aaa.bbb.ccc.ddd* Server, which generates responses and sends them back to the Load Balancer (or other testing device).

# Sample Balancer Configurations

Sample configuration:

- Router at `64.173.55.161` (netmask `255.255.255.224`), DNS server at `64.173.55.167`.

- 4 frontend Servers (`fe5`, `fe6`, `fe7`, `fe8`) with "real" IP addresses `64.173.55.{180,181,182,183}`
- inter-Cluster network `192.168.10.xxx`, with frontend "cluster" addresses `192.168.10.{5,6,7,8}`
- load balancer with `64.173.55.164` address (VIP address).
- a loopback interface on each frontend Server has an alias configured for the `64.173.55.164` address configured.

The multi-IP no-NAT RTP method is used.

The CommuniGate Pro configuration (WebAdmin Settings realm):

- the Network->LAN IP->Cluster-wide page: WAN IPv4 Address: `64.173.55.164`
- the Network->LAN IP->Server-wide page (on each frontend Server): WAN IPv4 Address: `64.173.55.{180,181,182,183}`
- the RealTime->SIP->Receiving->UDP Listener page (on each frontend Server): {port `5060`, address:`64.173.55.164`} and {port: `5060`, address: `all addresses`;}

A "no-NAT" configuration with "normal" load balancing for POP, IMAP, and "DSR" load balancing for SIP (UDP/TCP), SMTP, HTTP User (8100).

The Load Balancer configuration:

**Foundry ServerIron®** (`64.173.55.176` is its service address)

```
Startup configuration:
!
server predictor round-robin
!
server real fe5 64.173.55.180
 port pop3
 port pop3 keepalive
 port imap4
 port imap4 keepalive
 port 5060
 port 5060 keepalive
 port smtp
 port smtp keepalive
 port 8100
 port 8100 keepalive
!
server real fe6 64.173.55.181
 port pop3
 port pop3 keepalive
 port imap4
 port imap4 keepalive
 port 5060
 port 5060 keepalive
 port smtp
 port smtp keepalive
 port 8100
 port 8100 keepalive
!
server real fe7 64.173.55.182
 port pop3
 port pop3 keepalive
 port imap4
 port imap4 keepalive
 port 5060
 port 5060 keepalive
 port smtp
 port smtp keepalive
 port 8100
 port 8100 keepalive
!
server real fe8 64.173.55.183
 port pop3
 port pop3 keepalive
 port imap4
 port imap4 keepalive
 port 5060
 port 5060 keepalive
 port smtp
 port smtp keepalive
```

```
 port 8100
 port 8100 keepalive
!
!
server virtual vip1 64.173.55.164
 predictor round-robin
 port pop3
 port imap4
 port 5060
 port 5060 dsr
 port smtp
 port smtp dsr
 port 8100
 port 8100  dsr
 bind pop3  fe5 pop3  fe6 pop3  fe7 pop3  fe8 pop3
 bind imap4 fe5 imap4 fe6 imap4 fe7 imap4 fe8 imap4
 bind 5060  fe8 5060  fe7 5060  fe6 5060  fe5 5060
 bind smtp  fe8 smtp  fe7 smtp  fe6 smtp  fe5 smtp
 bind 8100  fe5 8100  fe6 8100  fe7 8100  fe8 8100
!
ip address 64.173.55.176 255.255.255.224
ip default-gateway 64.173.55.161
ip dns server-address 64.173.55.167
ip mu act
end
```

Note: you should NOT use the `port 5060 sip-switch`, `port sip sip-proxy-server`, or other "smart" (application-level) Load Balancer features.

**Alteon/Nortel AD3®** (64.173.55.176 is its service address, hardware port 1 is used for up-link, ports 5-8 connect frontend Servers)

```
script start "Alteon AD3" 4  /**** DO NOT EDIT THIS LINE!
/* Configuration dump taken 21:06:57 Mon Apr  9, 2007
/* Version 10.0.33.4,  Base MAC address 00:60:cf:41:f5:20
/c/sys
        tnet ena
        smtp "mail.communigate.com"
        mnet 64.173.55.160
        mmask 255.255.255.224
/c/sys/user
        admpw "ffe90d3859680828b6a4e6f39ad8abdace262413d5fe6d181d2d199b1aac22a6"
/c/ip/if 1
        ena
        addr 64.173.55.176
        mask 255.255.255.224
        broad 64.173.55.191
/c/ip/gw 1
        ena
        addr 64.173.55.161
/c/ip/dns
        prima 64.173.55.167
/c/sys/ntp
        on
        dlight ena
        server 64.173.55.167
/c/slb
        on
/c/slb/real 5
        ena
        rip 64.173.55.180
        addport 110
        addport 143
        addport 5060
        addport 25
        addport 8100
        submac ena
/c/slb/real 6
        ena
        rip 64.173.55.181
        addport 110
        addport 143
        addport 5060
        addport 25
        addport 8100
        submac ena
/c/slb/real 7
```

```
          ena
          rip 64.173.55.182
          addport 110
          addport 143
          addport 5060
          addport 25
          addport 8100
          submac ena
/c/slb/real 8
          ena
          rip 64.173.55.183
          addport 110
          addport 143
          addport 5060
          addport 25
          addport 8100
          submac ena
/c/slb/group 1
          add 5
          add 6
          add 7
          add 8
          name "all-services"
/c/slb/port 1
          client ena
/c/slb/port 5
          server ena
/c/slb/port 6
          server ena
/c/slb/port 7
          server ena
/c/slb/port 8
          server ena
/c/slb/virt 1
          ena
          vip 64.173.55.164
/c/slb/virt 1/service pop3
          group 1
/c/slb/virt 1/service imap4
          group 1
/c/slb/virt 1/service 5060
          group 1
          udp enabled
          udp stateless
          nonat ena
/c/slb/virt 1/service smtp
          group 1
          nonat ena
/c/slb/virt 1/service 8100
          group 1
          nonat ena
/
script end  /**** DO NOT EDIT THIS LINE!
```

**F5 Big-IP®** (`64.173.55.176` is its service address)

Use the *nPath Routing* feature for SIP UDP/TCP traffic. This is F5 Networks, Inc. term for the Direct Server Response method.

Because F5 BigIP is not a switch, you must use the DSR (nPath Routing) method for all services.
`bigip_base.conf:`

```
vlan external {
   tag 4093
   interfaces
      1.1
      1.2
}
stp instance 0 {
   vlans external
   interfaces
      1.1
         external path cost 20K
         internal path cost 20K
      1.2
         external path cost 20K
         internal path cost 20K
}
```

```
self allow {
    default
        udp snmp
        proto ospf
        tcp https
        udp domain
        tcp domain
        tcp ssh
}
self 64.173.55.176 {
    netmask 255.255.255.224
    vlan external
    allow all
}
```

bigip.conf:

```
partition Common {
    description "Repository for system objects and shared objects."
}
route default inet {
    gateway 64.173.55.161
}
monitor MySMTP {
    defaults from smtp
    dest *:smtp
    debug "no"
}
profile fastL4 CGS_fastL4 {
    defaults from fastL4
    idle timeout 60
    tcp handshake timeout 15
    tcp close timeout 60
    loose initiation disable
    loose close enable
    software syncookie disable
}
pool Frontends {
    monitor all MySMTP and gateway_icmp
    members
        64.173.55.180:any
        64.173.55.181:any
        64.173.55.182:any
        64.173.55.183:any
}
node * monitor MySMTP
```

bigip_local.conf:

```
virtual address 64.173.55.164 {
    floating disable
    unit 0
}
virtual External {
    translate address disable
    pool Frontends
    destination 64.173.55.164:any
    profiles CGS_fastL4
}
```

# Outgoing TCP Connections

When VIP addresses are assigned to CommuniGate Pro Domains, you may want to configure your CommuniGate Pro Modules to initiate outgoing TCP connections using these VIP addresses as source IP addresses. If you do so, the response TCP packets will be directed to the Load Balancer, which should be configured to direct them to the proper Cluster Member - to the CommuniGate Pro Server that has initiated the TCP connection.

For each Cluster Member that can initiate TCP connections (usually the frontend servers), select a port range for outgoing connections. These ranges should not intersect. For example, select the port range 33000-33999 for the first Cluster Member, 34000-34999 for the Cluster Member, etc.

Make sure that the server OS is configured so that the selected port range is outside of the OS "ephemeral port" range. For example, the following command can be used to check the Linux OS "ephemeral port" range:

```
[prompt]# cat /proc/sys/net/ipv4/ip_local_port_range
32768 61000
[prompt]#
```

and the following command can be used to change the Linux OS "ephemeral port" range:

```
[prompt]# echo "50000 61000" >/proc/sys/net/ipv4/ip_local_port_range
cat /proc/sys/net/ipv4/ip_local_port_range
50000 61000
[prompt]#
```

To make these changes permanent, add the following line to the Linux /etc/sysctl.conf file:

```
net.ipv4.ip_local_port_range = 50000 61000
```

For each of these Cluster members, open the Network settings in the WebAdmin Settings realm, and specify the selected TCP port range. Disable the Use for Media Proxy only option to make the CommuniGate Pro Server software use the selected port range for all outgoing TCP connections with a predefined source address.

Configure the Load Balancer: all packets coming to VIP address(es) and to any port in the selected port range should be directed to the corresponding Cluster Member.

# Software Load Balancer

The CommuniGate Pro Dynamic Cluster can be used to control software load balancers (such as Linux IPVS), running on the same systems as the Cluster members.

Select the cluster members to distribute the incoming traffic to. In a frontend-backend configuration, you would usually use all or some of the frontend servers for that.

Make sure that all selected cluster members have the VIP addresses configured as "loopback aliases" (see above).

Use the WebAdmin Interface to open the Cluster page in the Settings realm and select the Load Balancer group A for all selected servers:

| Load Balancer Group: | A | Balancer Weight: | 100 |
|---|---|---|---|

Balancer Weight
Use this setting to specify relative weight of this server in the Load Balancer Group. The more this value, the larger part of incoming TCP connections and UDP packets will be directed to this server.

All or some of the selected servers should be equipped with a software load balancer, and they should have an "External Load Balancer" Helper application configured. This application should implement the Load Balancer Helper protocol.

| External Load Balancer | | | Help |
|---|---|---|---|
| Log Level: | All Info | Program Path: | |
| Time-out: | 30 sec | Auto-Restart: | 60 sec |

As soon as the first Load Balancer helper application starts on some Cluster Member, the Cluster Controller activates that Helper, making it direct all incoming traffic to its Cluster member, and distribute that traffic to all active Cluster members in its Load Balancer Group.

If the Cluster Member running the active Load Balancer fails or it is switched into the "non-ready" state, the Cluster Controller activates some other Load Balancer member in that group (if it can find one).

## Linux IPVS

The CommuniGate Pro Linux package includes the `Services/IPVSHelper.sh` shell application that can be used to control the IPVS software load balancer.

The application expects that the VIP addresses are stored in the `/etc/sysconfig/vipaddrs` file, and the local interface (lo) aliases for these addresses have beeen created (see above).

Specify `$Services/IPVSHelper.sh` *parameters* as the External Load Balancer "program path", and start it by selecting the Helper checkbox.
The following parameters are supported:

`-p` *number*
  persistence: all connections from the same IP address will be directed to the same Cluster Member if they come within *number* seconds after the existing connection. Specify 0 (zero) to switch persistance off. The default value is 15 seconds.
`-i` *interface*
  the ethernet interface used to receive packets addressed to the VIP address(es). The default value is `eth0`
`-s` *number*
  the "syncID" value used to synchronize connection tables on the active load balancer and other Cluster Members that can become load balancers. The default value is 0.
`-t` *number*
  the time-out (in seconds) for reading a command sent by the CommuniGate Pro Server. The default value is 15.
`-f` *filePath*
  the file system path for the file containing the list of the VIP addresses. The default value is `/etc/sysconfig/vipaddrs`
`-r` *number*
  the relative weight of the active load balancer in the Load Balancer group. All other group members have the weight of 100. The default value is 100.
`-m`
  if this parameter is specified, the Helper application does not execute actual shell commands, but only copies the commands it would execute to its standard output, so they are recorded in the CommuniGate Pro System Log.

**Note:** the Linux kernel 3.5.3-1 or better is recommended. When an earlier version is used, all TCP connections made to the active Load Balancer are dropped, when a different server becomes the active Load Balancer.

**Note:** If a Cluster member has the External Balancer Helper application switched on, and then it is switched off, some active connections may be broken. If you do not plan to switch the Helper application back on, restart the `ipvsadm` service or switch it off completely.

Introduction

Installation

SysAdmin

Network

Objects

Storage

E-mail

Real-Time

Access

Services

Directory

Clusters

Applications

Miscellaneous

Licensing

WebMail

Pronto!

PBX

Applications | Data | CLI/API | Rules | CG/PL | PBXApp | XIMSS | WebApp | WSSP | Helpers

# Applications

- **Concepts**

Many CommuniGate Pro components can be customized or extended using various types of programming techniques. These sections provide the information about the basic CommuniGate Pro programming concepts, the data models used, and the Application Programming Interfaces (APIs) available.

# Concepts

Internally, the CommuniGate Pro Server software uses an object-oriented data model. The model includes "simple" objects (such as strings, numbers, datablocks, timestamps, and other "atomic" objects), as well as "structured" objects (such as arrays and dictionaries).
The same objects are used for CommuniGate Pro applications and APIs.
The Data Formats section describes these data objects, and specifies their textual representations.

All Administration (provisioning, management, tuning, monitoring) functions of the CommuniGate Pro Server are available via the Network CLI interface. This simple, text-based TCP protocol is used to integrate the CommuniGate Pro system with various external systems, including provisioning and billing.

As all other protocols and modules, the CLI module is supported with the CommuniGate Pro SSI (Single Service Image) infrastructure, so a single CLI connection established with any Cluster member can be used to manage and control the entire Cluster. Actually, the SSI component itself uses the same CLI protocol for inter-cluster communications.

Some installations may require non-standard, complex, and/or heavily customized feature sets. The CommuniGate Pro system acts as an Application Server platform, implementing a very simple, but effective high-level CG/PL programming language.
This language is used to create simple, yet powerful custom applications.
CG/PL applications extend the standard CommuniGate Pro feature set without performance and reliability degradation usually associated with third-party application platforms.
The same CG/PL language is used in different product components, the only difference is in the built-in function sets offered by each component.
The CG/PL language uses the same data model as the internal product components.

The CommuniGate Pro services may employ certain functions not directly implemented in the product itself. Content-filtering (Anti-virus and Anti-Spam) products, spell-checkers, billing processors, and many other products (or "engines") can be integrated with the CommuniGate Pro Server using the Helpers mechanism.

Applications **Data** CLI/API Rules CG/PL PBXApp XIMSS WebApp WSSP Helpers

# Data Formats

- **Textual Representation**
- **Atomic Objects**
  - **Strings**
  - **DataBlocks**
  - **Numbers**
  - **Time Stamps**
  - **IP Addresses**
  - **Null Object**
  - **Other Objects**
- **Structured Objects**
  - **Arrays**
  - **Dictionaries**
  - **XML**
- **Formal Syntax Rules**
- **XML Objects**
  - **Strings**
  - **DataBlocks**
  - **Numbers**
  - **Time Stamps**
  - **IP Addresses**
  - **Null Object**
  - **Arrays**
  - **Dictionaries**
  - `vCard`
  - `vCardGroup`
  - `iCalendar`
  - `vCalendar`
  - `vevent`, `vtodo`
  - `vfreebusy`
  - `xrule`
  - `x509`
  - `sdp`
- **ASN.1**

The CommuniGate Pro Server processes most types of data as generic *objects*. This section describes the generic, fundamental object types used in all Server components.

For each object type, a textual representation of the object data is specified.

When an object is stored in a file, sent in a [CLI/API](#) command response, or extracted from the Server in any other way, the object textual representation is used.

When the Server reads an object from outside (from a settings file, from a CLI command, etc.), it converts the provided textual representation into an internal object.

# Textual Representation

Objects can be represented as a single-line or multi-line text (depending on the context).

Object representation elements can be separated using "white spaces". Only the space symbol can be used as a "white space" in single-line texts.

In multi-line texts the space and "horizontal tab" symbols, as well as the EOL (End-Of-Line, CR and/or LF) symbols are treated as "white spaces". Additionally, a "comment" is treated as a "white space", where a comment consists of:

- two consecutive forward slash symbols (`//`) and all following symbols up to and including the first EOL symbol.
- the consecutive forward slash and asterisk symbols (`/*`) and all following symbols up to and including the consecutive asterisk and forward slash symbols (`*/`).

---

# Atomic Objects

Atomic objects are unstructured, generic data-type objects.

## Strings

A string is a sequence of UTF-8 encoded text bytes, not containing a binary zero byte.

A textual representation of a string is either an *atom* - a sequence of Latin letters (in the ASCII encoding) and digits, or a *quoted string* - a sequence of any printable symbols (using the UTF-8 encoding) except the quotation mark and the backslash symbol, enclosed into the quotation marks (`"`).

Examples:
```
MyName    My2ndName    "My Name with spaces and the . symbol"
```

If you want to include the quotation mark symbol into a quoted string, include the backslash symbol and the quotation mark, if you want to include the backslash symbol into a quoted string, include two backslash symbols.

Examples:
```
"a \"string\" within string"    "Single \\ backslash"
```

You can use the `\r` symbol combination to include the *carriage-return* symbol into a string, you can use the `\n` symbol combination to include the *line-feed* symbol into a string, and you can use the `\e` symbol combination to include the system-independent *End-Of-Line* symbol(s) into a string.

Examples:
```
"Line1\eLine2"    "TEXT3\rTEXT67\nTEXT78"
```

Use the `\r` or `\n` combinations to include the carriage-return and line-feed characters only when they are NOT used as line separators.

You can use the `\t` symbol combination to include the *tabulation* symbol into a string.

Example:
```
"Line1:\tField1\tField2\eLine2:\tField1\tField2"
```

You can use the `\nnn` symbol combination to include any symbol into a string, if *nnn* is a 3-digit decimal number equal to the ASCII code of the desired symbol.

Example:
```
"Using the \012 (Vertical Tabulation) symbol"
```

You can use the `\u'nnn'` symbol combination to include any Unicode symbol into a string, if *nnn* is a hexadecimal number equal to the Unicode code of the desired symbol.

Example:
```
"Using the \u'2764' (Heavy black heart) symbol"
```

In multi-line texts, a string can be represented as two or more consecutive strings with zero or more white spaces between them. This feature allows you to break very long string objects into several strings and put them on multiple text lines.

## DataBlocks

DataBlocks are basic, unstructured blocks of binary data. They are composed as text strings with the Base64 encoding of binary data enclosed into brackets.

Example:
`[HcqHfHI=]`
this is a binary block containing the following 5 binary data bytes: `0x1D 0xCA 0x87 0x7C 0x72`

White spaces can be placed before and after the brackets and between any symbols of the encoded data.

## Numbers

Numbers are basic, unstructured data objects. Each Number object contains one 64-bit signed integer value. A Number is presented as a text string starting with the `#` symbol, followed by an optional minus (`-`) symbol, followed by either 1 or more decimal digits, or by the *base-indicator* and 1 or more digits of the specified base.
The base indicator `0x` specifies the base value 16 (a hexadecimal number), the base indicator `0o` specifies the base value 8 (a octal number), the base indicator `0b` specifies the base value 2 (a binary number).

Examples:
```
#-234657
#0x17EF
#-0b1000111000
```

## Time Stamps

Time Stamps are basic, unstructured data objects. Each Time Stamp object contains one global time value. The time value is presented in GMT time, as a text string starting with the `#T` symbols, and containing the day, months, year, and, optionally, the hour, minute, and the second values.

Example:
```
#T22-10-2009_15:24:45
```

There are 2 special Time Stamps - for the "remote past" (`#TPAST`) and for the "remote future" (`#TFUTURE`).

## IP Addresses

IP Addresses are basic, unstructured data objects. Each IP Address object contains an IPv4 or IPv6 address, and, optionally a port number. The IP Address is presented as a text string starting with the `#I` symbols, and containing the canonical IPv4 or IPv6 address, optionally followed by the port number.

Examples:
```
#I[10.0.44.55]:25   #I[2001:470:1f01:2565::a:80f]:25
```

## Null Object

A null-object is a unique atomic object used to represent absence of any other object.
The Null Object is presented as the `#NULL#` text string.

## Other Objects

When the Server converts internal data objects into the text presentation, it may encounter some custom, application-specific objects. These objects are presented as `#(objectName:address)` text strings, where *objectName* is an alpha-numeric object class name, and *address* is a hexadecimal number - the object memory address.

These objects must NOT be used in text presentation being converted into a data object.

# Structured Objects

## Arrays

An array object is an ordered set of objects (array elements).

Array textual representation is a list of its element representations, separated with the comma (`,`) symbols, and enclosed into the parentheses.

Example:
```
(Element1 , "Element2" , "Element 3")
```

An array element can be any object - a string, an array, a dictionary, etc.

Example:
```
(Element1 , ("Sub Element1", SubElement2) , "Element 3")
```

Any number of spaces, tabulation symbols, and/or line breaks (end-of-line symbols) can be placed between a parenthesis and an element, and between an element and a comma symbol.

Example:
```
(
  Element1   ,
  (    "Sub Element1",
   SubElement2   )
  ,
"Element 3"   )
```

An array may have zero elements (an empty array).

Example:
```
()
```

## Dictionaries

A dictionary object is a set of key-value pairs. Dictionary keys are strings. Each key in a dictionary should be unique. The dictionary keys are processed as **case-sensitive** strings, unless explicitly specified otherwise.

Any object can be used as a value associated with a key.

A dictionary textual representation is a sequence of its key value pairs, enclosed into the curvy brackets.
Each pair is represented as its key string representation, followed by the equal (`=`) symbol, followed by the textual representation of the associated value object, followed by the semicolon (`;`) symbol.

Example:
```
{Key1=Element1; Key2 ="Element2" ; "Third Key"="Element 3"; }
```

The value object in any key-value pair can be a string, an array, a dictionary, or any other object.

Example:
```
{Key1=(Elem1,Elem2); Key2={Sub1="XXX 1"; Sub2=X245;}; }
```

Any number of spaces, tabulation symbols, and/or line breaks (end-of-line symbols) can be placed between a bracket and a pair, around the equal symbol, and around the semicolon symbol.

Example:
```
{
 Key1  =   (Elem1,Elem2)   ;
 Key2 = {   Sub1 = "XXX 1";
    Sub2=X245;   };
}
```

A dictionary may have zero elements (an empty dictionary).

Example:
```
{}
```

## XML

An XML object is an XML document. It has a name, a set of namespaces (strings), a set attributes with string values, and zero, one, or several body elements. Each body element is either a string or an XML object.

An XML object textual representation is its standard textual representation, starting with the angle bracket symbol.

# Formal Syntax Rules

Below is the formal syntax definition for textual representations of the basic type objects.

```
b-digit      ::= 0 | 1

o-digit      ::= 0 .. 7

d-digit      ::= 0 .. 9

h-digit      ::= d-digit | A .. F | a .. f

a-symbol     ::= A .. Z | a .. z | d-digit

u-symbol     ::= a-symbol | any non-ASCII utf-8 symbol

l-symbol     ::= u-symbol | . | - | _ | @

atom         ::= 1* l-symbol

b-symbol     ::= a-symbol | + | / | =

s-symbol     ::= any printable symbol except " and \ | \\ | \" | \r | \n | \e | \ d-digit d-digit d-digit

string       ::= " 0* s-symbol " | atom

datablock    ::= [ 1* b-symbol ]

day          ::= 0 .. 3 d-digit (2-digit number in the 1..31 range)

month        ::= 0 .. 1 d-digit (2-digit number in the 1..12 range)

year         ::= 1 .. 2 d-digit d-digit d-digit (4-digit number in the 1970..2038 range)

hour         ::= 0 .. 2 d-digit (2-digit number in the 0..23 range)

minute       ::= 0 .. 5 d-digit (2-digit number in the 0..59 range)

second       ::= 0 .. 5 d-digit (2-digit number in the 0..59 range)

number       ::= # [ -] 1* d-digit | # [ -] 0x 1* h-digit | # [ -] 0b 1* b-digit | # [ -] 0o 1* o-digit

timestamp    ::= # T day - month - year [ _ hour : minute : second ]

ip4          ::= 1* d-digit . 1* d-digit . 1* d-digit . 1* d-digit

ip6          ::= 0*(1* h-digit :) [: [ 0*(1* h-digit :) ] ]

ip-address   ::= # I[ [ip4 | ip6] ] [ : 1*d-digit ]

null-        ::= #NULL#
object

array        ::= ( [object 0*( , object ) ] )

dictionary   ::= { 0*( string = object ; ) }

XML          ::= <XML standard format >

object       ::= string | datablock | number | timestamp | ip-address | null-object | array | dictionary
                 | XML
```

# XML Objects

CommuniGate Pro can convert complex structures (such as vCards, iCalendar, SDP objects) into generic XML objects. An XML presentation may also be required for Objects such as datablocks, arrays, or dictionaries. This section specifies the CommuniGate Pro XML presentation for all these objects and complex structures.

## String

A string is presented as a text element.
If this is a top XML element, a string is presented as an `object` XML element, with the text body containing this string.

If a string contains special characters, it should be represented as a `binString` element with a text body containing the base64-encoded string.
An empty string is presented as an empty `binString` element.

## Datablock

A datablock is presented as a `base64` XML element.
The XML element body is the base64-encoded datablock content.

Example:
```
<base64>STYRyui=</base64>
```

## Number

A number is presented as a `number` XML element.
The XML element body is the text presentation of the number object value (a decimal value, or, if the base indicator is given, a hexadecimal, octal, or binary value).

Examples:
```
<number>123456</number>, <number>0x78FAB5</number>
```

## Time Stamp

A Time Stamp is presented as a `date` XML element.
The XML element body is the text presentation of the time stamp in the iCalendar format.

Example:
```
<date>20101122T123000Z</date>
```

## IP Addresses

An IP Address is presented as an `ipAddr` XML element.
The XML element body is the text presentation of the IP address, enclosed with square brackets, and optional decimal port number separated with the colon symbol.

Example:
```
<ipAddr>[10.0.2.2]:8010</ipAddr>
```

## Null-value

A null-value object is presented as an empty `null` XML element.

## Array

An array is presented as a sequence of one or more `subValue` XML elements.
The XML element body represents an Array element.
An empty array is presented as one `subValue` XML element without a body.
If this is a top XML element, the array is presented as an `object` XML element, with the text body containing `subValue` XML elements.

Example:
```
<subValue>my string</subValue><subValue><number>123456</number></subValue>
```

# Dictionary

A [dictionary](#) is presented as a sequence of one or more `subKey` XML elements.
The XML element `key` attribute presents the dictionary element key, and the XML body represents the dictionary element value.
An empty dictionary is presented as one `subKey` XML element without a `key` attribute and without a body.
If this is a top XML element, the dictionary is presented as an `object` XML element, with its body containing `subKey` XML elements.

Example:

```
<subKey key="firstKey">my string</subKey><subKey key="secondKey"><number>123456</number></subKey>
```

## vCard

This XML element represents a vCard object (as specified in the Jabber/XEP vCard XML documents).

Attributes:
`modified`
> This optional attribute contains the value of the `REV` property (iCalendar-formatted GMT time).

Body:
> Contains vCard properties as XML elements with the same names, converted to the uppercase ASCII.
> Each property element contains:

> Attributes:
> > *none*

> Body:
> > a set of XML sub-elements representing property parameters and the property value:

> > parameters
> > > XML elements with the same names as the parameter names, converted to the uppercase ASCII.
> > > The `ENCODING` and `QUOTED-PRINTABLE` vCard property parameters are used to decode the property value and they are not stored.
> > value

> > > *structured value* (`N`,`ORG`,`ADR`)
> > > > a set of XML elements with structure element names, and text bodies containing a subpart of the structured property value.
> > > *binary value*
> > > > a `BINVAL` XML element with a text body containing the base64-encoded property value.
> > > *text value*
> > > > a `VALUE` XML element with a text body containing the property value.

> Example:

```
begin:VCARD
source:ldap://cn=bjorn Jensen, o=university of Michigan, c=US
name:Bjorn Jensen
n:Jensen;bjorn;A;Mr;II
email;type=INTERNET:bjorn@umich.edu
org:U of Michigan;Computer Science Dept.
tel;type=WORK,MSG:+1 313 747-4454
key;type=x509;encoding=B:dGhpcyBjb3VsZCBiZSANKbXkgY2VydGlmaWNhdGUK
end:VCARD
```

```
<vCard>
  <SOURCE><VALUE>ldap://cn=bjorn Jensen, o=university of Michigan, c=US</VALUE></SOURCE>
  <NAME><VALUE>Bjorn Jensen</VALUE></NAME>
  <N><FAMILY>Jensen</FAMILY><GIVEN>bjorn</GIVEN>
    <MIDDLE>A</MIDDLE><PREFIX>Mr.</PREFIX><SUFFIX>II</SUFFIX></N>
  <EMAIL><VALUE>bjorn@umich.edu</VALUE></EMAIL>
  <ORG><ORGNAME>U of Michigan</ORGNAME><ORGUNIT>Computer Science Dept.</ORGUNIT></ORG>
  <TEL><WORK /><MSG /><VALUE>+1 313 747-4454</VALUE></TEL>
  <KEY><X509 /><BINVAL>dGhpcyBjb3VsZCBiZSANKbXkgY2VydGlmaWNhdGUK</BINVAL></KEY>
</vCard>
```

## vCardGroup

This XML element represents a vCardGroup object.

Attributes:
modified
    This optional attribute contains the value of the REV property (iCalendar-formatted GMT time).

Body:
    vCardGroup properties as XML elements with the same names, converted to the uppercase ASCII: NAME, NOTE, UID. Each element includes a VALUE XML element with a text body containing the property value.

    Zero, one, or several MEMBER XML Elements, one for each group member. The XML element text body is the group member address, while an optional CN attribute can contain the element "real name".

    Example:

```
BEGIN:VGROUP
PRODID:CommuniGate Pro 5.2
VERSION:1.0
NAME:Basket Buddies
NOTE:My basketball team.
UID:594895837.1@team.dom
REV:20071214T174114Z
MEMBER;CN="Jack Nimble":jack@nimble.dom
MEMBER;CN="Tim Slow":tim@slow.dom
END:VGROUP
```

```
<vCardGroup modified="20071214T174114Z">
   <NAME><VALUE>Basket Buddies</VALUE></NAME>
   <NOTE><VALUE>My basketball team.</VALUE></NOTE>
   <UID><VALUE>594895837.1@team.dom</VALUE></UID>
   <MEMBER CN="Jack Nimble">jack@nimble.dom</MEMBER>
   <MEMBER CN="Tim Slow">tim@slow.dom</MEMBER>
</vCardGroup>
```

## iCalendar

This XML element represents an iCalendar object.

Body:
    a vCalendar element.

## vCalendar

This XML element represents a vCalendar object.

Attributes:
version
    the vCalendar version (2.0 for iCalendar)
method
    an optional attribute with a vCalendar method
prodid
    an optional identification string of the product used to create this vCalendar object.

Body:
    a set of vtimezone, vevent, vtodo, vfreebusy elements.

## vtimezone

This XML element represents a VTIMEZONE object.

Body:
    a set of XML elements:

- tzid: the element body contains the zone name.
- standard and, optionally, daylight: these element bodies contain the following XML elements:
    - tzoffsetto: the element body contains the time shift value.
    - tzoffsetfrom (optional): the element body contains the time shift value.
        rrule

(optional): the element body contains a string with a recurrence descriptor.

Example:

```
BEGIN:VTIMEZONE
TZID:Europe/Central
BEGIN:STANDARD
DTSTART:19710101T030000
TZOFFSETFROM:+0200
TZOFFSETTO:+0100
RRULE:FREQ=YEARLY;BYMONTH=10;BYDAY=-1SU
END:STANDARD
BEGIN:DAYLIGHT
DTSTART:19710101T020000
TZOFFSETFROM:+0100
TZOFFSETTO:+0200
RRULE:FREQ=YEARLY;BYMONTH=3;BYDAY=-1SU
END:DAYLIGHT
END:VTIMEZONE
```

```
<vtimezone>
   <tzid>Europe/Central</tzid>
   <standard>
      <dtstart>19710101T030000</dtstart>
      <tzoffsetto>+0100</tzoffsetto>
      <rrule>FREQ=YEARLY;BYMONTH=10;BYDAY=-1SU</rrule>
   </standard>
   <daylight>
      <dtstart>19710101T020000</dtstart>
      <tzoffsetto>+0200</tzoffsetto>
      <rrule>FREQ=YEARLY;BYMONTH=3;BYDAY=-1SU</rrule>
   </daylight>
</vtimezone>
```

**vevent, vtodo**

These XML elements represent VEVENT and VTODO objects.

Attributes:

`localTime`

> this attribute is present for non-recurrent objects only. It contains the date and time the object starts. The time is the local time in the Time Zone selected for the current user.

`localStart`

> this attribute is present for recurrent objects only. It contains the time the object starts (seconds since midnight). The time is the local time in the Time Zone selected for the current user.

Body:

> a set of XML elements, each representing one property.
> The property parameters are represented as element attributes.
> The property value is represented as the element body.
> If the value starts with the `MAILTO:` prefix, this prefix is removed.
> If an object has "exceptions", the `exceptions` XML element is added. Its XML sub-elements represent the exception VEVENT or VTODO objects.

Example:

```
BEGIN:VEVENT
ORGANIZER;CN="Jim Smith":MAILTO:jim_smith@example.com
RRULE:FREQ=WEEKLY;INTERVAL=2;BYDAY=MO,TU,WE,TH;UNTIL=20060305T000000Z
DTSTAMP:20051204T140844Z
UID:566852630.4@mail.example.com
SEQUENCE:1
SUMMARY:test - recurrent
DTSTART;TZID=NorthAmerica/Pacific:20051204T100000
DTEND;TZID=NorthAmerica/Pacific:20051204T110000
X-MICROSOFT-CDO-BUSYSTATUS:BUSY
LAST-MODIFIED:20051204T140844Z
CREATED:20051204T140844Z
PRIORITY:5
END:VEVENT
BEGIN:VEVENT
UID:566852630.4@mail.example.com
RECURRENCE-ID:20051206T180000Z
SUMMARY:test - recurrent (later this time)
DTSTART;TZID=NorthAmerica/Pacific:20051206T120000
DTEND;TZID=NorthAmerica/Pacific:20051206T130000
END:VEVENT
```

```
<vevent>
   <organizer CN="Jim Smith">jim_smith@example.com</organizer>
   <rrule>FREQ=WEEKLY;INTERVAL=2;BYDAY=MO,TU,WE,TH;UNTIL=20060305T000000Z</rrule>
```

```
    <dtstamp>20051204T140844Z</dtstamp>
    <uid>566852630.4@mail.example.com</uid>
    <sequence>1</sequence>
    <summary>test - recurrent</summary>
    <dtstart tzid="NorthAmerica/Pacific">20051204T100000</dtstart>
    <dtend tzid="NorthAmerica/Pacific">20051204T110000</dtend>
    <busystatus>BUSY</busystatus>
    <last-modified>20051204T140844Z</last-modified>
    <created>20051204T140844Z</created>
    <priority>5</priority>
    <exceptions>
        <uid>566852630.4@mail.example.com</uid>
        <recurrenceId>20051206T180000Z</recurrenceId>
        <summary>test - recurrent (later this time)</summary>
        <dtstart tzid="NorthAmerica/Pacific">20051206T120000</dtstart>
        <dtend tzid="NorthAmerica/Pacific">20051206T130000</dtend>
    </exceptions>
</vevent>
```

**vfreebusy**

These XML elements represent VFREEBUSY objects.

Attributes:
dtstart
> the start time of the time period covered with this VFREEBUSY object.

dtend
> the end time of the time period covered with this VFREEBUSY object.

dtstamp
> the time when this VFREEBUSY object was composed.

Body:
> a set of `freebusy` XML elements, each representing one time interval. Time intervals are not intersecting.

> Attributes:
> dtstart
> > the start time of the time interval.

> dtend
> > the end time of the time interval.

> fbtype
> > the optional busy-type status. If not specified, the time interval has the BUSY status.

Example:
```
BEGIN:VFREEBUSY
DTSTART:20080329T075517Z
DTEND:20080604T075517Z
DTSTAMP:20080405T075517Z
FREEBUSY:20080329T075517Z/20080329T120000Z
FREEBUSY:20080330T070000Z/20080330T120000Z
FREEBUSY:20080331T070000Z/20080331T120000Z
END:VFREEBUSY
```

```
<vfreebusy dtend="20080604T075517Z" dtstamp="20080405T075517Z" dtstart="20080329T075517Z">
    <freebusy dtend="20080329T120000Z" dtstart="20080329T075517Z" />
    <freebusy dtend="20080330T120000Z" dtstart="20080330T070000Z" />
    <freebusy dtend="20080331T120000Z" dtstart="20080331T070000Z" />
</vfreebusy>
```

**xrule**

This XML element represents an recurrence object.

Attributes:
freq
> the recurrence type.

interval
> the interval parameter. If absent, interval is assumed to be 1.

wkst
> the week start day name. If absent, the MO (Monday) is assumed.

count
> an optional integer attribute specifying the COUNT parameter.

until

an optional attribute specifying the UNTIL parameter in the iCalendar date-time format.

Body:

a set of XML elements:

BYMONTH, BYWEEKNO, BYYEARDAY, BYMONTHDAY, BYSETPOS
> each element body contains a number - the month number, the week number, etc. Except for the BYMONTH element, the number can be negative.

BYDAY
> each element body contains one day name (MO, TU, .. SU); each element can contain an optional numeric week attribute - the week number (it can be negative).

Example:

```
RRULE:FREQ=WEEKLY;INTERVAL=2;BYDAY=MO,TU,WE,TH;UNTIL=20060305T000000Z
```

```
<xrule freq="WEEKLY" interval="2" until="20060305T000000Z">
  <BYDAY>MO</BYDAY><BYDAY>TU</BYDAY><BYDAY>WE</BYDAY><BYDAY>TH</BYDAY>
</xrule>
```

Example:

```
RRULE:FREQ=YEARLY;BYMONTH=10;BYDAY=-1SU
```

```
<xrule freq="YEARLY">
  <BYMONTH>10</BYMONTH><BYDAY week="-1">SU</BYDAY>
</xrule>
```

## x509

This XML element represents an X.509 Certificate.

Attributes:

version
> the Certificate format version.

subject
> optional; an E-mail address associated with this Certificate.

Body:

a set of XML elements:

subject,issuer
> these XML elements contain XML sub-elements for the name parts of the Certificate Subject and Certificate Issuer. Supported sub-elements include c, country, l, city, etc. The sub-element value is the textual body of the sub-element.

validFrom,validTill
> a textual body of each of these elements is the valifity date-time presented in the iCalendar format

base64
> an optional element - its body is the base64-encoded certificate data.

## sdp

This XML element represents an SDP object.

Attributes:

ip
> the default media IP address (optional).

origUser
> the username field of the session origin (optional).

sessionID
> the sess-id field of the session origin.

sessionVersion
> the sess-version field of the session origin.

origIP
> the IP address specified in the nettype, addrtype, and unicast-address fields of the session origin.

subject
> the session subject (optional). If missing, the - subject is used.

Body:
 a set of *info*, `attribute`, and/or `media` elements.

**media**

This XML element represents an SDP Media object.

Attributes:
`media`
 the media type (such as `audio`, `video`). If this attribute is missing, the `audio` value is used.
`ip`
 the media address and port.
`protocol`
 the media protocol (such as `udp`, `tcp`, `RTP/AVP`). If this attribute is missing, the `RTP/AVP` value is used.
`direction`
 the media direction (`sendrecv`, `sendonly`, `recvonly`, `inactive`).
`rtcp`
 the rtcp address and port (optional)

Body:
 a set of `codec`, *info*, and `attribute` elements.

**codec**

This XML element represents an SDP Media codec.

Attributes:
`id`
 the codec ID - the RTP payload number (a number in the 0..127 range).
`name`
 the codec name (such as `PCMU/8000`).
`format`
 the codec format parameter.

**attr**

This XML element represents an SDP or SDP Media attribute.

Attributes:
`name`
 the attribute name.

Body:
 optional: an attribute value string.

*info*

This XML element represents an SDP or SDP Media information field element. Supported elements are:
for SDP: `info`, `uri`, `email`, `phone`, `phone`, `bandwidth`, `time`, `repeat`, `zone`, and `key` elements;
for SDP Media: `title`, `bandwidth`, and `key` elements;

Attributes:
 *none*.

Body:
 optional: a field value string.

Example: a sample SDP document and its XML presentation:

```
v=0
o=- 6385718 9999 IN IP4 192.168.1.65
s=-
```

```
e=support@communigate.com
c=IN IP4 192.168.1.65
t=0 0
a=sdpattr1:sdpvalue
a=sdpattr2
m=audio 16398 RTP/AVP 0 4 8 101
t=title value
a=rtpmap:0 PCMU/8000
a=rtpmap:4 G723/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=sendrecv
a=mediaattr1:mediavalue
a=mediaattr2
```

```
<sdp ip="[192.168.1.65]" origUser="-" sessionID="6385718" sessionVersion="9999" originIP="[192.168.1.65]">
   <email>support@communigate.com</email>
   <attr name="sdpattr1">sdpvalue</attr>
   <attr name="sdpattr2" />
   <media media="audio" ip="[192.168.1.65]:16398" protocol="RTP/AVP" direction="sendrecv">
     <codec id="0" name="PCMU/8000" />
     <codec id="4" name="G723/8000" />
     <codec id="8" name="PCMA/8000" />
     <codec id="101" name="telephone-event/8000" format="0-15" />
     <title>title value</title>
     <attr name="mediaattr1">mediavalue</attr>
     <attr name="mediaattr2" />
   </media>
</sdp>
```

# ASN.1

CommuniGate Pro can convert ASN.1 structures into generic objects.

Each ASN.1 data element is converted into an [array](#).

For ASN.1 data elements of the Universal class, the array element number 0 is a string with the data element type: `boolean`, `integer`, `bits`, `octets`, `null`, `oid`, `objdescr`, `external`, `real`, `enum`, `embed`, `string`, `reloid`, `rsrv1`, `rsrv2`, `seq`, `set`, `stringNum`, `stringPrint`, `stringT61`, `stringIA5`, `string22`, `timeUTC`, `time`, `stringGraphic`, `stringISO64`, `stringGeneral`, `stringUniversal`, `string29`, `stringBMP`, `string31`.
Otherwise the array element number 0 is a string with the data element class (`application`, `choice`,`private`), and the array element number 1 is a number - the data element type code.

For "constructed" data elements, the remaining zero or more array elements contain converted ASN.1 data sub-elements.
For "primitive" data elements, the remaining array element contain the the data element presentation, which depends on the data element type:

`boolean`
    a [Null object](#) for "false", any other object for "true"
`integer`, `enum` (not more than 8 bytes long)
    a [Number](#)
`null`
    a [Null object](#)
`string`, *subtype*`string` (the data does not contain a binary zero)
    a [String](#)
`time`, `timeUTC`
    a [Time Stamp](#)
`oid`, `reloid`
    a [String](#) representing the ObjectID (`nn.mm.ll.kk.rr`)
*all other data types*
    a [Datablock](#)

---

Introduction

Installation

SysAdmin

Network

Objects

Storage

E-mail

Real-Time

Access

Services

Directory

Clusters

Applications

Miscellaneous

Licensing

WebMail

Pronto!

PBX

# Command Line Interface/API

- **CLI Access**
- **CLI Syntax**
- **Domain Set Administration**
- **Domain Administration**
- **Account Administration**
- **Group Administration**
- **Forwarder Administration**
- **Named Task Administration**
- **Access Rights Administration**
- **Mailbox Administration**
- **Alert Administration**
- **File Storage Administration**
- **Mailing Lists Administration**
- **Web Skins Administration**
- **Web Interface Integration**
- **Real-Time Application Administration**
- **Real-Time Application Control**
- **Account Services**
- **Server Settings**
- **Monitoring**
- **Statistics**
- **Directory Administration**
- **Miscellaneous Commands**
- **Index**

The CommuniGate Pro Server provides a Command Line Interface (CLI) for Server administrating. This interface can be used as an alternative for the Web Administrator interface.

CLI can also be used as the Application Program Interface (API), so the Server can be managed via scripts and other programs that issue the CLI commands to the Server.

# CLI Access

The CommuniGate Pro Server provides several methods to access its CLI:

- using TCP connections to the **PWD** module (as an extension to the PWD/poppwd protocol)
- using **CG/PL** `ExecuteCLI` command.
- using the **HTTP** CLI realm.
- using **XIMSS** `cliExecute` operation.

When the CLI is used over the PWD connection, the CLI commands are accepted as soon as the user is authenticated. For each CLI command the Server checks the authenticated user access rights.

If a command produces some data, the data is sent after the protocol line with the positive response. The CR-LF combination is sent after the data.

Below is a sample PWD session with CLI commands:

```
C: telnet servername.com 106
S: 200 CommuniGate Pro at mail.servername.com PWD Server 5.3 ready
C: USER postmaster
```

```
S:  300 please send the PASS
C:  PASS postmasterpassword
S:  200 login OK
C:  CreateAccount "user1"
S:  200 OK
C:  CreateAccount "user1"
S:  501 Account with this name already exists
C:  RenameAccount "user1" into "user2"
S:  200 OK
C:  CreateDomain "client1.com"
S:  200 OK
C:  CreateAccount "user1@client1.com" TextMailbox
S:  200 OK
C:  ListDomains
S:  200 data follow
S:  (mail.servername.com, client1.com, client2.com)
C:  QUIT
S:  200 CommuniGate Pro PWD connection closed
```

The CommuniGate Pro Perl Interface document contains a set of the Perl language utilities that allow a Perl script to access the CommuniGate Pro CLI API via the PWD protocol. The document also contains links to several useful sample Perl scripts (automated Account registration and removal, etc.)

The CommuniGate Pro Java Interface document contains the set of the Java language classes that allow a Java program to access the CommuniGate Pro CLI API via the PWD protocol. The document also contains links to several sample Java programs.

# CLI Syntax

The CommuniGate Pro CLI uses the standard Data Formats to parse commands and to format the output results.

**Note:** These Dictionary format syntax rules allow you to specify a string without the quotation marks if the string contains alphanumerical symbols only. You should use the quotation marks if a string contains the dot (.), comma (,), and other non-alphanumerical symbols.

In spite of the fact that the standard Data formats can use several text lines, all data (including arrays and dictionaries) you specify as CLI parameters should be stored on one command line.

If a CLI command produces some output in the array or dictionary format, the output data can be presented on several lines.

# Domain Set Administration

A user should have the All Domains Server access right to use the Domain Set Administration CLI commands.

The following commands are available for the System Administrators only:

LISTDOMAINS
> Use this command to get the list of domains. The command produces output data - an *array* with the names of all server domains.

MAINDOMAINNAME
> Use this command to get the name of the Main Domain. The command produces output data - a *string* with the Main Domain name.

GETDOMAINDEFAULTS
> Use this command to get the server-wide default Domain Settings. The command produces an output - a *dictionary* with the default Domain Settings.

UPDATEDOMAINDEFAULTS *newSettings*
> Use this command to change the server-wide default Domain settings.
>
> > *newSettings* : *dictionary*
> > This dictionary is used to update the default Domain settings dictionary. It does not have to contain all settings data, the omitted settings will be left unmodified.

SETDOMAINDEFAULTS *newSettings*

Use this command to change the server-wide default Domain settings.

> *newSettings* : *dictionary*
> This dictionary is used to replace the server-wide default Domain settings dictionary.

GETCLUSTERDOMAINDEFAULTS
UPDATECLUSTERDOMAINDEFAULTS *newSettings*
SETCLUSTERDOMAINDEFAULTS *newSettings*
> These commands are available in the Dynamic Cluster only.
> Use these commands instead of the [GET|UPDATE|SET]DOMAINDEFAULTS commands to work with the cluster-wide default Domain Settings.

GETSERVERACCOUNTDEFAULTS
> Use this command to get the server-wide Default Account settings. The command produces an output - a *dictionary* with the global default Account settings.

UPDATESERVERACCOUNTDEFAULTS *newSettings*
> Use this command to update the server-wide Default Account settings.

> *newSettings* : *dictionary*
> This dictionary is used to update the Default Account settings dictionary. It does not have to contain all settings data, the omitted settings will be left unmodified.

SETSERVERACCOUNTDEFAULTS *newSettings*
> Use this command to set the server-wide Default Account settings.

> *newSettings* : *dictionary*
> This dictionary is used to replace the server-wide Default Account settings dictionary.

GETCLUSTERACCOUNTDEFAULTS
UPDATECLUSTERACCOUNTDEFAULTS *newSettings*
SETCLUSTERACCOUNTDEFAULTS *newSettings*
> These commands are available in the Dynamic Cluster only.
> Use these commands instead of the [GET|UPDATE|SET]SERVERACCOUNTDEFAULTS commands to work with the cluster-wide Default Account settings.

GETSERVERACCOUNTPREFS
> Use this command to get the server-wide Default Account Preferences. The command produces an output - a *dictionary* with the default Preferences.

SETSERVERACCOUNTPREFS *newSettings*
> Use this command to change the server-wide Default Account Preferences.

> *newSettings* : *dictionary*
> This dictionary is used to replace the server-wide Default Account Preferences. All old server-wide Default Account Preferences are removed.

UPDATESERVERACCOUNTPREFS *newSettings*
> Use this command to change the server-wide Default Account Preferences.

> *newSettings* : *dictionary*
> This dictionary is used to update the Default Account Preferences. It does not have to contain all preferences data, the omitted Preferences will be left unmodified.

GETCLUSTERACCOUNTPREFS
SETCLUSTERACCOUNTPREFS *newSettings*

`UPDATECLUSTERACCOUNTPREFS` *`newSettings`*

These commands are available in the Dynamic Cluster only.

Use these commands instead of the [GET|SET|UPDATE]SERVERACCOUNTPREFS commands to work with the cluster-wide Default Account Preferences.

`CREATEDOMAIN` *`domainName`* `[ SHARED ] [ PATH` *`storage`* `] [` *`settings`* `]`

Use this command to create a new secondary Domain.

> *`domainName`* `:` *`string`*
>
> This parameter specifies the Domain name to create.
>
> *`storage`* `:` *`string`*
>
> This optional parameter specifies the "storage mount Point" directory for the Domain data (the name should be specified without the .mnt suffix).
>
> *`settings`* `:` *`dictionary`*
>
> This optional parameter specifies the Domain settings.

Use the SHARED keyword to create a Cluster-wide Domain in a Dynamic Cluster.

`RENAMEDOMAIN` *`oldDomainName`* `INTO` *`newDomainName`* `[ PATH` *`storage`* `]`

Use this command to rename a Domain.

> *`oldDomainName`* `:` *`string`*
>
> This parameter specifies the name of an existing secondary Domain.
>
> *`newDomainName`* `:` *`string`*
>
> This parameter specifies the new Domain name.
>
> *`storage`* `:` *`string`*
>
> This optional parameter specifies the new "storage mount Point" directory for the Domain data (the name should be specified without the .mnt suffix).

`DELETEDOMAIN` *`domainName`* `[ FORCE ]`

Use this command to remove a Domain.

> *`domainName`* `:` *`string`*
>
> This parameter specifies the name of the Domain to be removed.
>
> FORCE
>
> This optional parameter specifies that the Domain should be removed even if it is not empty. All Domain objects (Accounts, Groups, etc.) will be removed.

`CREATEDIRECTORYDOMAIN` *`domainName`* `[` *`settings`* `]`

Use this command to create a new directory-based Domain.

> *`domainName`* `:` *`string`*
>
> This parameter specifies the Domain name to create.
>
> *`settings`* `:` *`dictionary`*
>
> This optional parameter specifies the Domain settings.

This operation is allowed only when the Directory-based Domains are enabled.

`RELOADDIRECTORYDOMAINS`

Use this command to tell the server to scan the Domains Directory subtree so it can find all additional Directory-based Domains created directly in the Directory, bypassing the CommuniGate Pro Server.

This operation is allowed only when the Directory-based Domains are enabled.

`LISTSERVERTELNUMS` `[ FILTER` *`filter`* `]` *`limit`*

Use this command to read Telnum numbers created in all (non-clustered) Domains. The command produces an output - a

*dictionary* where each key is a Telnum number, and its value is the Account name it is assigned to. An numeric element for an empty (`""`) key is added, it contains the total number of Telnum numbers created.

    `filter` : `string`
        If this optional parameter is specified, only the telnum numbers containing the specified string are returned.

    `limit` : `number`
        The maximum number of Telnum numbers to return.

`LISTCLUSTERTELNUMS [ FILTER filter ] limit`
    The same as LISTSERVERTELNUMS, but for shared Cluster Domains.

`GETSERVERTRUSTEDCERTS`
    Use this command to get the server-wide set of Trusted Certificates. The command produces an output - an *array* of *datablocks*. Each *datablock* contains one X.509 certificate data.

`SETSERVERTRUSTEDCERTS newCertificates`
    Use this command to set the server-wide set of Trusted Certificates.

    `newCertificates` : `array`
        This array should contain *datablocks* with X.509 certificate data. It is used to replace the server-wide list of Trusted Certificates.

`GETCLUSTERTRUSTEDCERTS`
`SETCLUSTERTRUSTEDCERTS newCertificates`
    These commands are available in the Dynamic Cluster only.
    Use these commands instead of the [GET|SET]SERVERTRUSTEDCERTS commands to work with the cluster-wide set of Trusted Certificates.

`GETDIRECTORYINTEGRATION`
    Use this command to get the server-wide Directory Integration settings. The command produces an output - a *dictionary* with the Directory Integration settings.

`SETDIRECTORYINTEGRATION newSettings`
    Use this command to set the server-wide Directory Integration settings.

    `newSettings` : `dictionary`
        This dictionary is used to replace the server-wide Directory Integration settings dictionary.

`GETCLUSTERDIRECTORYINTEGRATION`
`SETCLUSTERDIRECTORYINTEGRATION newSettings`
    These commands are available in the Dynamic Cluster only.
    Use these commands instead of the [GET|SET]DIRECTORYINTEGRATION commands to work with the cluster-wide Directory Integration settings.

`CREATEDOMAINSTORAGE [ SHARED ] PATH storage`
    Use this command to create a "storage mount point" for new Domains.

    `storage` : `string`
        This parameter specifies the "storage mount Point" name.

    Use the SHARED keyword to create a "storage mount point" for Cluster Domains in a Dynamic Cluster.

`LISTDOMAINSTORAGE [ SHARED ]`
    Use this command to list "storage mount points" for Domains.
    The command produces an output - an *array* with "storage mount points" names.
    Use the SHARED keyword to list "storage mount point" for Cluster Domains in a Dynamic Cluster.

# Domain Administration

A user should have the [All Domains](#) Server access right or the [Domain Administration access right](#) to use the Domain Administration CLI commands.

GETDOMAINSETTINGS [ *domainName* ]
> Use this command to get the Domain settings. The command produces an output - a *dictionary* with the domainName settings. Only the explicitly set (not the default) settings are included into that dictionary.
>
> *domainName* : *string*
>> This optional parameter specifies the name of an existing Domain.

GETDOMAINEFFECTIVESETTINGS [ *domainName* ]
> Use this command to get the Domain settings. The command produces an output - a *dictionary* with the domainName settings. Both the explicitly set and the default settings are included into that dictionary.
>
> *domainName* : *string*
>> This optional parameter specifies the name of an existing Domain.

UPDATEDOMAINSETTINGS [ *domainName* ] *newSettings*
> Use this command to update the Domain settings.
>
> *domainName* : *string*
>> This optional parameter specifies the name of an existing Domain.
>
> *newSettings* : *dictionary*
>> This dictionary is used to update the Domain settings dictionary. It does not have to contain all settings data, the omitted settings will be left unmodified. If a new setting value is specified as the string default, the Domain setting value is removed, so the default Domain settings value will be used.
>
> If this command is used by a Domain Administrator, it will update only those Domain Settings that this Domain Administrator is allowed to modify.

GETACCOUNTDEFAULTS [ *domainName* ]
> Use this command to get the default Account settings for the specified Domain. The command produces an output - a *dictionary* with the default settings.
>
> *domainName* : *string*
>> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the Administrator Domain.

UPDATEACCOUNTDEFAULTS [ *domainName* ] *newSettings*
> Use this command to modify the Default Account settings for the specified Domain.
>
> *domainName* : *string*
>> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain.
>
> *newSettings* : *dictionary*
>> This dictionary is used to modify the Domain Default Account settings. The dictionary does not have to contain all settings data, the omitted settings will be left unmodified. If a new setting value is specified as the string default, the setting value is removed, so the global Server Default Account Settings will be used.
>
> If this command is used by a Domain Administrator, it will update only those Default Account settings this Administrator is allowed to modify.

GETACCOUNTDEFAULTPREFS [ *domainName* ]
> Use this command to get the Default Account Preferences for the specified Domain. The command produces an output - a *dictionary* with the default Preferences.

> > *domainName* : *string*
> > > This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain.

SETACCOUNTDEFAULTPREFS [ *domainName* ] *newSettings*
> Use this command to change the Default Account Preferences for the specified Domain.

> > *domainName* : *string*
> > > This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the authenticated user Domain.

> > *newSettings* : *dictionary*
> > > This dictionary is used to replace the Default Account Preferences. All old Default Account Preferences are removed.

> This command can be used by Domain Administrators only if they have the WebUserSettings access right.

UPDATEACCOUNTDEFAULTPREFS [ *domainName* ] *newSettings*
> Use this command to change the Default Account Preferences for the specified Domain.

> > *domainName* : *string*
> > > This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the authenticated user Domain.

> > *newSettings* : *dictionary*
> > > This dictionary is used to modify the Domain Default Account Preferences. It does not have to contain all Preferences data, the omitted elements will be left unmodified.
> > > If a new element value is specified as the string default, the Default Preferences value is removed, so the default Server-wide (or Cluster-wide) Account Preferences value will be used.

> This command can be used by Domain Administrators only if they have the WebUserSettings access right.

GETACCOUNTTEMPLATE [ *domainName* ]
> Use this command to get the Account Template settings. The command produces an output - a *dictionary* with the Template settings.

> > *domainName* : *string*
> > > This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain.

UPDATEACCOUNTTEMPLATE [ *domainName* ] *newSettings*
> Use this command to modify the Account Template settings.

> > *domainName* : *string*
> > > This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain.

> > *newSettings* : *dictionary*
> > > This dictionary is used to modify the Domain Account Template. All new Accounts in the specified Domain will be created with the Template settings. The dictionary does not have to contain all settings data, the omitted settings will be left unmodified. If a new setting value is specified as the string default, the Template setting value is removed.

> If this command is used by a Domain administrator, it will update only those Template settings that the Domain administrator is allowed to modify.

GETDOMAINALIASES *domainName*

Use this command to get the list of Domain Aliases. The command produces an output - an *array* with the Domain alias names.

> *domainName* : *string*
>> This parameter specifies the name of an existing Domain.

GETDOMAINMAILRULES *domainName*
> Use this command to get the list of Domain Queue Rules. The command produces an output - an *array* of the Queue Rules specified for the Domain.

> *domainName* : *string*
>> This parameter specifies the name of an existing Domain.

SETDOMAINMAILRULES *domainName newRules*
> Use this command to set the Domain Queue Rules.

> *domainName* : *string*
>> This parameter specifies the name of an existing Domain.

> *newRules* : *array*
>> This array should contain the Domain Queue Rules. All old Domain Queue Rules are removed.

> This command can be used by Domain Administrators only if they have the `RulesAllowed` access right.

GETDOMAINSIGNALRULES *domainName*
> Use this command to get the list of Domain Signal Rules. The command produces an output - an *array* of the Signal Rules specified for the Domain.

> *domainName* : *string*
>> This parameter specifies the name of an existing Domain.

SETDOMAINSIGNALRULES *domainName newRules*
> Use this command to set the Domain Signal Rules.

> *domainName* : *string*
>> This parameter specifies the name of an existing Domain.

> *newRules* : *array*
>> This array should contain the Domain Signal Rules. All old Domain Signal Rules are removed.

> This command can be used by Domain Administrators only if they have the `SignalRulesAllowed` access right.

LISTADMINDOMAINS [ *domainName* ]
> Use this command to get the list of Domains that can be administered by Domain Administrator Accounts in the specified domainName Domain. The command produces an output - an *array* with the Domain names.

> *domainName* : *string*
>> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the authenticated user Domain.

LISTDOMAINOBJECTS *domainName* [ FILTER *filter* ] *limit* [ ACCOUNTS ] [ ALIASES ] [ FORWARDERS ] [COOKIE *cookie*
> Use this command to get a list of Domain objects.

> *domainName* : *string*
>> This parameter specifies the Domain name.

> *filter* : *string*

This optional parameter specifies a filter string: only objects with names including this string as a substring are listed.

*limit* : *numeric string*
> This parameter specifies the maximum number of objects to list.

ACCOUNTS, ALIASES, FORWARDERS
> These keywords specify which Domain objects should be listed.

*cookie* : *string*
> This optional parameter specifies a "cookie" string.

The command produces output data - an array with the following elements:

- a numeric *string* with the total number of Domain Accounts
- a *dictionary* with Domain Objects. Each dictionary key is a Domain Object name. The dictionary value depends on the Domain Object type:

  Account
  > the dictionary object is a string (the Account file extension)

  Account Alias
  > the dictionary object is an *array*. Its only element is a string with the Alias owner (Account) name.

  Forwarder
  > the dictionary object is an *array*. Its only element is an *array*. Its only element is a string with the Forwarder address.

- a numeric *string* with the total number of Aliases in the Domain.
- a numeric *string* with the total number of Forwarders in the Domain.
- a new "cookie" *string* (optional, exists only if there was the COOKIE *cookie* part in the command.)

To list Objects in large Domains, specify some reasonable *limit* value (below 10,000) and specify and empty cookie string. If not all Objects are returned, issue the command again, using the new cookie value specified in the response array. When all Objects are returned, the new cookie value in the response is an empty string.

LISTACCOUNTS [ *domainName* ]
> Use this command to get the list of all Accounts in the Domain. The command produces output data - a *dictionary* with the keys listing all Accounts in the specified (or default) Domain.

*domainName* : *string*
> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain.

LISTDOMAINTELNUMS *domainName* [ FILTER *filter* ] *limit*
> Use this command to read Telnum numbers created in the specified Domain. The command produces an output - a *dictionary* where each key is a Telnum number, and its value is the Account name it is assigned to. An numeric element for an empty ("") key is added, it contains the total number of Telnum numbers created.

*domainName* : *string*
> This parameter specifies the Domain name.

*filter* : *string*
> If this optional parameter is specified, only the telnum numbers containing the specified string are returned.

*limit* : *number*
> The maximum number of Telnum numbers to return.

INSERTDIRECTORYRECORDS *domainName*

Use this command to insert records for Domain objects (Accounts, Groups, Mailing Lists, Forwarders) into the Directory.

> *domainName* : *string*
>> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the authenticated user Domain.

This command can be used by Domain Administrators only if they have the `CentralDirectory` access right.

DELETEDIRECTORYRECORDS *domainName*
> Use this command to delete Domain object records from the Directory.

> *domainName* : *string*
>> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the authenticated user Domain.

This command can be used by Domain Administrators only if they have the `CentralDirectory` access right.

CREATEACCOUNTSTORAGE *domainName* PATH *storage*
> Use this command to create a "storage mount point" for new Accounts in the Domain.

> *domainName* : *string*
>> This parameter specifies the Domain name.

> *storage* : *string*
>> This parameter specifies the "storage mount Point" name.

LISTACCOUNTSTORAGE *domainName*
> Use this command to list Account "storage mount points" in the specified Domain.
> The command produces an output - an *array* with "storage mount points" names.

> *domainName* : *string*
>> This parameter specifies the Domain name.

The following commands are available for the System Administrators only:

SETDOMAINALIASES *domainName* *newAliases*
> Use this command to set the Domain aliases.

> *domainName* : *string*
>> This parameter specifies the name of an existing Domain.

> *newAliases* : *array*
>> This array should contain the Domain alias name strings. All old Domain aliases are removed.

SETDOMAINSETTINGS *domainName* *newSettings*
> Use this command to change the Domain settings.

> *domainName* : *string*
>> This parameter specifies the name of an existing Domain.

> *newSettings* : *dictionary*
>> This dictionary is used to replace the Domain settings dictionary. All old Domain settings are removed.

SETACCOUNTDEFAULTS [ *domainName* ] *newSettings*
> Use this command to change the Default Account settings for the specified Domain.

> *domainName* : *string*
>> This parameter specifies the Domain name.
>
> *newSettings* : *dictionary*
>> This dictionary is used to replace the Domain Default Account settings. All old Account Default settings are removed.


SETACCOUNTTEMPLATE [ *domainName* ] *newSettings*
> Use this command to change the Account Template settings.
>
> *domainName* : *string*
>> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain.
>
> *newSettings* : *dictionary*
>> This dictionary is used to update the Domain Account Template. All new Accounts in the specified Domain will be created with the Template settings. All old Account Template settings are removed.


GETDOMAINLOCATION [ *domainName* ]
> Use this command to get the Domain file directory path (relative to the Server *base directory*). The command produces an output - a *string* with the Domain file path.
>
> *domainName* : *string*
>> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain.


SUSPENDDOMAIN *domainName*
> Use this command to suspend a Domain, so all currently active Accounts are closed and no Account can be opened in this Domain.
>
> *domainName* : *string*
>> This parameter specifies the name of the Domain to be suspended.


RESUMEDOMAIN *domainName*
> Use this command to resume a Domain, so Accounts can be opened in this Domain.
>
> *domainName* : *string*
>> This parameter specifies the name of the Domain to be resumed.

---

# Account Administration

A user should have the [All Domains](#) Server access right or the [Domain Administration access right](#) to use the Account Administration CLI commands.

CREATEACCOUNT *accountName* [ *accountType* ] [ PATH *storage* ] [ LEGACY ] [ *settings* ]
> Use this command to create new accounts.
>
> *accountName* : *string*
>> This parameter specifies the name for the new Account.
>> The name can contain the `@` symbol followed by the Domain name, in this case the Account is created in the specified Domain. If the Domain name is not specified, the command applies to the administrator Domain.
>
> *accountType* : MultiMailbox | TextMailbox | MailDirMailbox | SlicedMailbox | AGrade | BGrade | CGrade
>> This optional parameter specifies the type of the Account to create. If no Account type is specified a `MultiMailbox-`

type Account is created.

> *storage* : *string*
>> This optional parameter specifies the "storage mount Point" directory for the Account data (the name should be specified without the .mnt suffix).

> LEGACY
>> This optional flag tells the system to create an Account with a Legacy (visible for legacy mailers) INBOX.

> *settings* : *dictionary*
>> This optional parameter specifies the initial Account settings. Account is created using the settings specified in the Account Template for the target Domain. If the *settings* parameter is specified, it is used to modify the Template settings.

> This command can be used by Domain Administrators only if they have the `CanCreateAccounts` access right. Additionally, if a single-mailbox Account format is requested or the LEGACY flag is used, the Domain Administrators must have the `CanCreateSpecialAccounts` access right.
> If this command is used by a Domain Administrator, it will use only those Account settings this Administrator is allowed to modify.

RENAMEACCOUNT *oldAccountName* into *newAccountName* [ PATH *storage* ]
> Use this command to rename Accounts.

> *oldAccountName* : *string*
>> This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

> *newAccountName* : *string*
>> This parameter specifies the new Account name. The name can include the Domain name (see above).

> *storage* : *string*
>> This optional parameter specifies the "storage mount Point" directory for the moved Account data (the name should be specified without the .mnt suffix).

> This command can be used by Domain Administrators only if they have the `CanCreateAccounts` access right.

DELETEACCOUNT *oldAccountName*
> Use this command to remove Accounts.

> *oldAccountName* : *string*
>> This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

> This command can be used by Domain Administrators only if they have the `CanCreateAccounts` access right.

SETACCOUNTTYPE *accountName* *accountType*
> Use this command to change the Account type.

> *accountName* : *string*
>> This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

> *accountType* : MultiMailbox | AGrade | BGrade | CGrade
>> This parameter specifies the new Account type. The current Account type must also belong to this type set.

> This command can be used by Domain Administrators only if they have the `CanCreateAccounts` access right.

GETACCOUNTSETTINGS *accountName*
> Use this command to get the Account settings. The command produces an output - a *dictionary* with the Account settings. Only the explicitly set (not the default) Account settings are included into the dictionary.

> *accountName* : *string*
>> This parameter specifies the name of an existing Account. The name can include the Domain name (see above). You can also specify the single asterisk (∗) symbol instead of an Account name. This will indicate the current

authenticated Account.

**Note:** All users can send the GETACCOUNTSETTINGS command for their own Accounts.

UPDATEACCOUNTSETTINGS *accountName newSettings*
Use this command to update the Account settings.

> *accountName* : *string*
> This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

> *newSettings* : *dictionary*
> This dictionary is used to update the Account settings dictionary. It does not have to contain all settings data, the omitted settings will be left unmodified. If a new setting value is specified as the string default, the Account setting value is removed, so the default Account setting value will be used.

If this command is used by a Domain Administrator, it will update only those Account settings this Administrator is allowed to modify.

GETACCOUNTEFFECTIVESETTINGS *accountName*
Use this command to get the effective Account settings. The command produces an output - a *dictionary* with the Account settings. Both the explicitly set and the default Account settings are included into the dictionary.

> *accountName* : *string*
> This parameter specifies the name of an existing Account. The name can include the Domain name (see above). You can also specify the single asterisk (*) symbol instead of an Account name. This will indicate the current authenticated Account.

**Note:** All users can send the GETACCOUNTEFFECTIVESETTINGS command for their own Accounts.

GETACCOUNTONESETTING *accountName keyName*
Use this command to get a single setting from the effective Account settings list. The command produces an output - an *object* which can be a *string*, an *array* or a *dictionary* with the Account setting, or *null-object*.

> *accountName* : *string*
> This parameter specifies the name of an existing Account. The name can include the Domain name (see above). You can also specify the single asterisk (*) symbol instead of an Account name. This will indicate the current authenticated Account.

> *keyName* : *string*
> This parameter specifies the name of the setting to read.

**Note:** All users can send the GETACCOUNTONESETTING command for their own Accounts.

SETACCOUNTPASSWORD *accountName* PASSWORD *newPassword* [ METHOD *method* | NAME *tag* ] [ CHECK ]
Use this command to update the Account password.

> *accountName* : *string*
> This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

> *newPassword* : *string*
> This string specifies the new Account password. The new password will be stored using the effective Password Encryption setting of the target Account.

> *tag* : *string*
> This optional parameter specifies the tag for an application-specific password. If the *newPassword* string is empty, the corresponding application-specific password is removed.

> *method* : *string*
> This optional parameter specifies the Account Access Mode. If this mode is "SIP", the the Alternative SIP Password Setting is modified, if this mode is RADIUS, then the Alternative RADIUS Password Setting is modified. In all other cases, the CommuniGate Password setting is modified. The new password will be stored using the effective Password

Encryption setting of the target Account.

To use this command, the user should have the "Basic Settings" Domain Administration right for the target Account Domain.
Any user can modify her own Account password. In this case, or when the CHECK keyword is explicitly specified, the operation succeeds only if the the supplied password matches the size and complexity restrictions and the Account CanModifyPassword effective Setting is enabled.

VERIFYACCOUNTPASSWORD *accountName* PASSWORD *password*
Use this command to verify the Account password.

> *accountName* : *string*
> This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

> *password* : *string*
> This string is used to specify the password to check (in the clear text format).

To use this command, the user should have any Domain Administration right for the target Account Domain.

VERIFYACCOUNTIDENTITY *accountName* FOR *identity*
Use this command to check if the value of 'From:' header is allowed to be used by the Account.

> *accountName* : *string*
> This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

> *identity* : *string*
> This string is to be the value of 'From:' header, e.g. "Real Name <user@domain.dom>".

To use this command, the user should have any Domain Administration right for the target Account Domain.

GETACCOUNTALIASES *accountName*
Use this command to get the list of Account aliases. The command produces an output - an *array* with the Account alias names.

> *accountName* : *string*
> This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

SETACCOUNTALIASES *accountName* *newAliases*
Use this command to set the Account aliases.

> *accountName* : *string*
> This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

> *newAliases* : *array*
> This array should contain the Account alias name strings. All old Account aliases are removed.

This command can be used by Domain Administrators only if they have the CanCreateAliases access right.

GETACCOUNTTELNUMS *accountName*
Use this command to get the list of telephone numbers assigned to the Account. The command produces an output - an *array* with the assigned numbers.

> *accountName* : *string*
> This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

SETACCOUNTTELNUMS *accountName* *newTelnums*
Use this command to assign telephone numbers to the Account.

> *accountName* : *string*

This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

*newTelnums* : *array*
This array should contain the telephone number strings. All old numbers assigned to the Account are removed.

This command can be used by Domain Administrators only if they have the `CanCreateTelnums` access right.

MODIFYACCOUNTTELNUMS *accountName parameters*
Use this command to change telephone numbers assigned to the Account.

*accountName* : *string*
This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

*parameters* : *dictionary*
This dictionary should contain the `op` string element specifying the requested operation:

add
the *parameters* dictionary must contain the `telnum` string element with a telnum number to be added (atomically) to the set of Telnums assigned to the specified Account. If this set already contains this Telnum, an error code is returned.

del
the *parameters* dictionary must contain the `telnum` string element with a telnum number to be removed (atomically) from the set of Telnums assigned to the specified Account. If this set does not contain this Telnum, an error code is returned.

pop
The *parameters* dictionary must not contain any other elements. The first Telnum assigned to the specified Account is atomically removed from the Account Telnum set, and copied into the command result dictionary. If the Account Telnum set was empty, no error code is returned, and no element is copied into the command result dictionary.

The command produces an output - a *dictionary*. For the `pop` operation, this dictionary can contain the `telnum` string element - the Telnum removed from the Account Telnum set.
This command can be used by Domain Administrators only if they have the `CanCreateTelnums` access right.

GETACCOUNTMAILRULES *accountName*
Use this command to get the list of Account Queue Rules. The command produces an output - an *array* of the Queue Rules specified for the Account.

*accountName* : *string*
This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

SETACCOUNTMAILRULES *accountName newRules*
Use this command to set the Account Queue Rules.

*accountName* : *string*
This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

*newRules* : *array*
This array should contain the Account Queue Rules. All old Account Queue Rules are removed.

This command can be used by Domain Administrators only if they have the `RulesAllowed` access right.
This command can be used by any Account user to modify own Rules (subject to "allowed actions" restrictions).

GETACCOUNTSIGNALRULES *accountName*
Use this command to get the list of Account Signal Rules. The command produces an output - an *array* of the Signal Rules specified for the Account.

accountName : *string*
>   This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

SETACCOUNTSIGNALRULES *accountName* *newRules*
>   Use this command to set the Account Signal Rules.

>   accountName : *string*
>   >   This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

>   newRules : *array*
>   >   This array should contain the Account Signal Rules. All old Account Signal Rules are removed.

>   This command can be used by Domain Administrators only if they have the `SignalRulesAllowed` access right.

UPDATEACCOUNTMAILRULE *accountName* *newRule*
UPDATEACCOUNTMAILRULE *accountName* DELETE *oldRule*
UPDATEACCOUNTSIGNALRULE *accountName* *newRule*
UPDATEACCOUNTSIGNALRULE *accountName* DELETE *oldRule*
>   Use these commands to update an Account Queue or Signal Rule.

>   accountName : *string*
>   >   This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

>   newRule : *array*
>   >   This parameter should be an array, its first element specifies the Rule priority, its second element specifies the Rule name. The optional third, forth, and fifth elements specify the Rule conditions, Rule actions, and Rule comment.
>   >   If the parameter array contains less than 4 elements, the array first element is used to update the priority of the already existing Rule with the name specified as the second array element. If such a Rule does not exist, the command returns an error.
>   >   If the parameter array contains 4 or more elements, the entire parameter array is stored as a new Rule. If there is an existing Rule with the same name, it is removed.

>   oldRule : *string*
>   >   This string parameter (specified after the DELETE keyword) specifies a name of the Rule to be removed. If such a Rule does not exist, the command does nothing and it does not return an error.

>   The `UpdateAccountMailRule` command can be used by Domain Administrators only if they have the `RulesAllowed` access right.
>   The `UpdateAccountSignalRule` command can be used by Domain Administrators only if they have the `SignalRulesAllowed` access right.
>   This command can be used by any Account user to modify own Rules (subject to "allowed actions" restrictions).

GETACCOUNTRPOPS *accountName*
>   Use this command to get the list of Account RPOP records. The command produces an output - a *dictionary* with RPOP records specified for the Account.

>   accountName : *string*
>   >   This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

SETACCOUNTRPOPS *accountName* *newRecords*
>   Use this command to set the Account RPOP records.

>   accountName : *string*
>   >   This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

>   newRecords : *dictionary*
>   >   This dictionary should contain the Account RPOP records. All old Account RPOP records are removed.

This command can be used by Domain Administrators only if they have the `CanModifyRPOP` access right.

GETACCOUNTRSIPS *accountName*
> Use this command to get the list of Account RSIP records. The command produces an output - a *dictionary* with RSIP records specified for the Account.
>
> *accountName* : *string*
>> This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

SETACCOUNTRSIPS *accountName newRecords*
> Use this command to set the Account RSIP records.
>
> *accountName* : *string*
>> This parameter specifies the name of an existing Account. The name can include the Domain name (see above).
>
> *newRecords* : *dictionary*
>> This dictionary should contain the Account RSIP records. All old Account RSIP records are removed.

This command can be used by Domain Administrators only if they have the `CanModifyRSIP` access right.

UPDATESCHEDULEDTASK *accountName taskData*
> Use this command to set the Account Scheduled Task records.
>
> *accountName* : *string*
>> This parameter specifies the name of an existing Account. The name can include the Domain name (see above).
>
> *taskData* : *dictionary*
>> This dictionary should contain the Scheduled Task data:
>>
>> id
>>> the Scheduled Task name string. If there is no existing task with this name, a new Scheduled Task record is created.
>>
>> program
>>> the Scheduled Task program name string. It should be a name of the [Real-Time Application](#) available for the Account Domain environment. If this element is not specfied, an existing Scheduled Task record (if any) is deleted.
>>
>> parameter
>>> an optional [simple Object](#). When the Scheduled Task program is launched, this Object is passed to it as its `startParameter` element.
>>
>> when
>>> a timestamp (GMT time) specifying when the Scheduled Task should be launched, or `now` string.
>>
>> period
>>> an optional parameter - a `day`, `week`, `month`, or `year` string, or a number. When specified, the Scheduled Task is automatically re-scheduled after the specified period of time (if this parameter is a number, then it specified the number of seconds).
>>> If this parameter is not specified, the Scheduled Task record is removed as soon as the Task is launched.

> When a Scheduled Task is launched, its `main` entry point is launched. The Task `startParameter` array contains the following elements:
>
> * `startParameter[0]` is the Scheduled Task name string
> * `startParameter[1]` is the timestamp specifying the moment the Task was started
> * `startParameter[2]` (optional) is the Scheduled Task `parameter` data

This command can be used by Domain Administrators with the `CanModifyRSIP` access right for the target Account.

GETACCOUNTRIGHTS *accountName*
>    Use this command to get the array of the Server or Domain access rights granted to the specified user. The command produces output data - an *array* listing all Account Server Access rights.
>
>    *accountName* : *string*
>    >    This parameter specifies the name of an existing Account. The name can include the Domain name.


GETACCOUNTINFO *accountName* [Key *keyName* | ( *keyList*) ]
>    Use this command to get an element of the Account "info" dictionary. The command produces an output - see below.
>
>    *accountName* : *string*
>    >    This parameter specifies the name of an existing Account. The name can include the Domain name (see above). You can also specify the single asterisk (*) symbol instead of an Account name. This will indicate the current authenticated Account.
>
>    *keyList* : *array*
>    >    This optional parameter specifies the names of the info keys to retrieve.
>    >    Note that when Account "info" data are stored in .info dictionary files, the "info" elements have dictionary names starting with the hash (#) symbol. You should NOT include the hash symbol into the keyName parameter of the GETACCOUNTINFO command.
>    >    Sample:
>    >    <span style="color:blue">GETACCOUNTINFO "user1@domain1.com" (LastLogin,LastAddress)</span>
>    >    **Note:** the "info" element names are case-sensitive.
>    >    The output is a *dictionary* with all those "info" elements that exist and are specified in the keyList array.
>
>    *keyName* : *string*
>    >    This optional parameter specifies the name of the requested "info" element. It can be specified only if the *keyList* parameter is not specified.
>    >    Note that when Account "info" data are stored in .info dictionary files, the "info" elements have dictionary names starting with the hash symbol. You should NOT include the hash symbol into the keyName parameter of the GETACCOUNTINFO command.
>    >    Sample:
>    >    <span style="color:blue">GETACCOUNTINFO "user1@domain1.com" Key LastLogin</span>
>    >    **Note:** the "info" element names are case-sensitive.
>    >    The output is the specified "info" element. If the element is not found, the output is an empty string - two quotation marks ("").


>    **Note:** All users can use the GETACCOUNTINFO command to retrieve elements from their own Account "info" data.

GETACCOUNTPREFS *accountName*
>    Use this command to get the Account Preferences. The command produces an output - a *dictionary* with the Account Preferences.
>
>    *accountName* : *string*
>    >    This parameter specifies the name of an existing Account. The name can include the Domain name (see above).


>    **Note:** All users can use the GETACCOUNTPREFS command to retrieve their own Account Preferences.

UPDATEACCOUNTPREFS *accountName* *newSettings*
>    Use this command to modify the Account Preferences.
>
>    *accountName* : *string*
>    >    This parameter specifies the name of an existing Account. The name can include the Domain name (see above).
>
>    *newSettings* : *dictionary*
>    >    This dictionary is used to update the Account Preferences dictionary. It does not have to contain all Preferences data, the omitted elements will be left unmodified. If a new Preferences value is specified as the string default, the Preferences value is removed, so the default Preferences value will be used.


>    This command can be used by Domain Administrators only if they have the WebUserSettings access right.

`SETACCOUNTPREFS` *accountName newSettings*
> Use this command to set the Account Preferences.
>
> *accountName* : *string*
> > This parameter specifies the name of an existing Account. The name can include the Domain name (see above).
>
> *newSettings* : *dictionary*
> > This dictionary should contain the new Account Preferences. All old Account Preferences are removed.
>
> This command can be used by Domain Administrators only if they have the `WebUserSettings` access right.

`GETACCOUNTEFFECTIVEPREFS` *accountName*
> Use this command to get the effective Account Preferences. The command produces an output - a *dictionary* with Account Preferences. Both the explicitly set and the default settings are included into that dictionary.
>
> *accountName* : *string*
> > This parameter specifies the name of an existing Account. The name can include the Domain name (see above).
>
> **Note:** All users can use this command to retrieve their own effective Preferences.

`KILLACCOUNTSESSIONS` *accountName*
> Use this command to interrupt all Account sessions (POP, IMAP, FTP, WebUser, etc.).
>
> *accountName* : *string*
> > This parameter specifies the name of an existing Account. The name can include the Domain name (see above).
>
> **Note:** All Domain Administrators can use this command.

The following command manage the Account [Access Rights](). These command can be used by the Account owner and by Domain Administrators who have the `CanImpersonate` access right.

`GETACCOUNTACL` *accountName* [ AUTH *authAccountName* ]
> Use this command to get the Account Rights ACLs (Access Control Lists). The command produces an output - a *dictionary* with the ACL elements.
>
> *accountName* : *string*
> > This parameter specifies the name of an existing Account (target Account). The asterisk (*) symbol can be used to specify the current authenticated Account.
>
> *authAccountName* : *string*
> > This optional parameter specifies the name of an Account on whose behalf the operation should be executed. If this name is specified, the ACL info is returned only if the specified Account has the `Admin` access right for the target Account.

`SETACCOUNTACL` *accountName* [ AUTH *authAccountName* ] *newACL*
> Use this command to modify the access control list for the Account Access Rights.
>
> *accountName* : *string*
> > This parameter specifies the name of an existing Account (target Account). The asterisk (*) symbol can be used to specify the current authenticated Account.
>
> *authAccountName* : *string*
> > This optional parameter specifies the name of an Account on whose behalf the operation should be executed. If this name is specified, the ACL info is updated only if the specified Account has the `Admin` access right for target Account.
>
> *newACL* : *dictionary*
> > This parameter specifies the access right elements to be modified. Each dictionary key specifies an *identifier*, and the key value should be a string with access right symbols.

If the key value string starts with the minus ("-") symbol, access rights specified in the string are removed from the access right element.

If the key value string starts with the plus ("+") symbol, access rights specified in the string are added to the access right element.

In other cases, access rights specified in the string replace the set of rights in the access right element.

If the access right element for the specified key did not exist, it is created.

If the new access right element has empty set of access rights, the element is removed.

`GETACCOUNTACLRIGHTS` *accountName* `AUTH` *authAccountName*

This command produces an output - a *string* with the effective access rights for the given authAccountName.

*accountName* : *string*

This parameter specifies the name of an existing Account (target Account). The asterisk (*) symbol can be used to specify the current authenticated Account.

*authAccountName* : *string*

This parameter specifies the name of an Account whose effective access rights for the target Account should be retrieved.

The following commands are available for the System Administrators only:

`SETACCOUNTSETTINGS` *accountName* *newSettings*

Use this command to change the Account settings.

*accountName* : *string*

This parameter specifies the name of an existing Account.

*newSettings* : *dictionary*

This dictionary is used to replace the Account settings dictionary. All old Account settings are removed.

`GETACCOUNTLOCATION` *accountName*

Use this command to get the Account file directory path (for multi-mailbox Accounts) or the Account INBOX Mailbox path (for single-mailbox Accounts). The command produces an output - a *string* with the Account file path. The path is relative to the file directory of the Account Domain.

*accountName* : *string*

This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

`GETACCOUNTPRESENCE` *accountName*

Use this command to get the Account "presence" status. The command produces an output:

- *array* of two *strings* - the Account "presence" status and its custom status message, or
- *string* - the Account "presence" status (if no custom status message is set), or
- null-object - if the Account "presence" status is not set at all.

*accountName* : *string*

This parameter specifies the name of an existing Account. The name can include the Domain name (see above).

The `AccessMode` Account and Domain Setting specifies [Enabled Services](). The Setting value can be one of the following:

- The `All` string: all services are enabled.
- The `None` string: all services are disabled.
- An array of strings. The first array element is a number or a numeric string, the other array elements are names of the

enabled services.

All services with numbers larger than the value of the first array element are enabled, too.

The currently supported services (with their numbers) are:

1:`Mail`, 2:`POP`, 3:`IMAP`, 4:`WebMail`, 5:`PWD`, 6:`ACAP`, 7:`WebSite`, 8:`Relay`, 9:`Mobile`, 10:`FTP`, 11:`MAPI`, 12:`TLS`, 13:`S/MIME`, 14:`LDAP`, 15:`WebCAL`, 16:`RADIUS`, 17:`SIP`, 18:`PBX`, 19:`XMPP`, 20:`XIMSS`, 21:`Signal`, 22:`AirSync`, 23:`HTTP`, 24:`MobilePBX`, 25:`XMedia`, 26:`YMedia`, 27:`MobilePronto`

---

# Group Administration

A user should have the [All Domains](#) Server access right or the [Domain Administration access right](#) to use the Groups Administration CLI commands.

`LISTGROUPS` [ *domainName* ]

Use this command to get the list of all Groups in the Domain. The command produces output data - an *array* with the names of all Groups in the specified (or default) Domain.

*domainName* : *string*

This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain.

`CREATEGROUP` *groupName* [ *settings* ]

Use this command to create new Groups.

*groupName* : *string*

This parameter specifies the name for the new Group.

The name can contain the `@` symbol followed by the Domain name, in this case the Group is created in the specified Domain. If the Domain name is not specified, the command applies to the administrator Domain.

*settings* : *dictionary*

This optional parameter specifies the initial Group settings and the members list.

This command can be used by Domain Administrators only if they have the `CanCreateGroups` access right.

`RENAMEGROUP` *oldGroupName* into *newGroupName*

Use this command to rename Groups.

*oldGroupName* : *string*

This parameter specifies the name of an existing Group. The name can include the Domain name (see above).

*newGroupName* : *string*

This parameter specifies the new Group name. The name can include the Domain name (see above).

This command can be used by Domain Administrators only if they have the `CanCreateGroups` access right.

`DELETEGROUP` *groupName*

Use this command to remove Groups.

*groupName* : *string*

This parameter specifies the name of an existing Group. The name can include the Domain name (see above).

This command can be used by Domain Administrators only if they have the `CanCreateGroups` access right.

`GETGROUP` *groupName*

Use this command to get the Group settings. The command produces an output - a *dictionary* with the Group settings and members.

*groupName* : *string*

This parameter specifies the name of an existing Group. The name can include the Domain name (see above).

```
SETGROUP groupName newSettings
```
Use this command to set the Group settings.

```
groupName : string
```
This parameter specifies the name of an existing Group. The name can include the Domain name (see above).

```
newSettings : dictionary
```
This dictionary is used to replace the Group settings dictionary.

This command can be used by Domain Administrators only if they have the `CanCreateGroups` access right.

---

# Forwarder Administration

A user should have the [All Domains](#) Server access right or the [Domain Administration access right](#) to use the Forwarders Administration CLI commands.

```
LISTFORWARDERS [ domainName ]
```
Use this command to get the list of all Forwarders in the Domain. The command produces output data - an *array* with the names of all Forwarders in the specified (or default) Domain.

```
domainName : string
```
This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain.

```
CREATEFORWARDER forwarderName TO address
```
Use this command to create new Forwarders.

```
forwarderName : string
```
This parameter specifies the name for the new Forwarder.
The name can contain the `@` symbol followed by the Domain name, in this case the Forwarder is created in the specified Domain. If the Domain name is not specified, the command applies to the administrator Domain.

```
address : string
```
This parameter specifies the E-mail address the Forwarder should reroute E-mail messages and Signals to.

This command can be used by Domain Administrators only if they have the `CanCreateForwarders` access right.

```
RENAMEFORWARDER oldForwarderName INTO newForwarderName
```
Use this command to rename Forwarders.

```
oldForwarderName : string
```
This parameter specifies the name of an existing Forwarder. The name can include the Domain name (see above).

```
newForwarderName : string
```
This parameter specifies the new Forwarder name. The name can include the Domain name (see above).

This command can be used by Domain Administrators only if they have the `CanCreateForwarders` access right.

```
DELETEFORWARDER forwarderName
```
Use this command to remove Forwarders.

```
forwarderName : string
```
This parameter specifies the name of an existing Forwarder. The name can include the Domain name (see above).

This command can be used by Domain Administrators only if they have the `CanCreateForwarders` access right.

`GETFORWARDER` *`forwarderName`*
> Use this command to get the Forwarder address. The command produces an output - a *string* with the E-mail address this Forwarder reroutes all E-mail messages and Signals to.

> *`forwarderName`* : *`string`*
>> This parameter specifies the name of an existing Forwarder. The name can include the Domain name (see above).


`FINDFORWARDERS` *`domainName`* `TO` *`forwarderAddress`*
> Use this command to find all Forwarders pointing to the specified address. The command produces an output - an *array* with the found Forwarder names.

> *`domainName`* : *`string`*
>> This parameter specifies the Domain name.

> *`forwarderAddress`* : *`string`*
>> This parameter specifies an E-mail address to look for.


# Named Task Administration

A user should have the [All Domains](#) Server access right or the [Domain Administration access right](#) to use the Named Task Administration CLI commands.

`LISTDOMAINNAMEDTASKS` `[` *`domainName`* `]`
> Use this command to get the list of all Named Tasks in the Domain. The command produces output data - a *dictionary* where the keys are the Named Task names, and the values are dictionaries, containing the Task owner name, the task Real Name, and the name of the Real-Time Application program this Named Task runs.

> *`domainName`* : *`string`*
>> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain.


`LISTACCOUNTNAMEDTASKS` *`accountName`*
> Use this command to get the list of all Named Tasks owned by the specified Account. The command produces output data - a *dictionary* containing the same data as the LISTDOMAINNAMEDTASKS command result.

> *`accountName`* : *`string`*
>> This parameter specifies the owner Account name.


`CREATENAMEDTASK` *`taskName`* `FOR` *`accountName`*
> Use this command to create new Named Tasks.

> *`taskName`* : *`string`*
>> This parameter specifies the name for the new Named Task.
>> The name can contain the `@` symbol followed by the Domain name, in this case the Named Task is created in the specified Domain. If the Domain name is not specified, the command applies to the administrator Domain.

> *`accountName`* : *`string`*
>> This parameter specifies the owner Account name. It must not contain the `@` symbol and a Domain name, as this owner Account must be in the same Domain as the Named Task itself.


> This command can be used by Domain Administrators only if they have the `CanCreateNamedTasks` access right.

`RENAMENAMEDTASK` *`oldTaskName`* `into` *`newTaskName`*

Use this command to rename Named Tasks.

> *oldTaskName* : *string*
> This parameter specifies the name of an existing Named Task. The name can include the Domain name (see above).

> *newTaskName* : *string*
> This parameter specifies the new Named Task name.

This command can be used by Domain Administrators only if they have the `CanCreateNamedTasks` access right.

DELETENAMEDTASK *taskName*
Use this command to remove Named Tasks.

> *taskName* : *string*
> This parameter specifies the name of an existing Named Task. The name can include the Domain name (see above).

This command can be used by Domain Administrators only if they have the `CanCreateNamedTasks` access right.

GETNAMEDTASK *taskName*
Use this command to get the Named Task settings. The command produces an output - a *dictionary* with the Named Task settings.

> *taskName* : *string*
> This parameter specifies the name of an existing Named Task. The name can include the Domain name (see above).

UPDATENAMEDTASK *taskName newSettings*
Use this command to set the Named Task settings.

> *taskName* : *string*
> This parameter specifies the name of an existing Named Task. The name can include the Domain name (see above).

> *newSettings* : *dictionary*
> This dictionary is used to update the Named Task settings dictionary.

This command can be used by Domain Administrators only if they have the `CanCreateNamedTasks` access right.

---

# Access Rights Administration

A user should have the [Master](#) Server access right to use the Access Rights Administration CLI commands.

SETACCOUNTRIGHTS *accountName newRights*
Use this command to set the Account Server Access rights.

> *accountName* : *string*
> This parameter specifies the name of an existing Account. The name can include the Domain name.

> *newRights* : *array*
> This array should contain the Access Right codes. All old Account access rights are removed.

To set access rights for an Account in a secondary Domain (i.e. Domain Administration Rights), the user may have only the [All Domains](#) Server access right.

---

# Mailbox Administration

A user should be the Mailbox owner, or should have the [All Domains](#) Server access right or the `CanAccessMailboxes` [Domain Administration](#) access right to use the Mailbox Administration CLI commands.

LISTMAILBOXES *accountName* [ FILTER *filter* ] [ AUTH *authAccountName* ]

Use this command to get the list of Account Mailboxes. The command produces an output - a *dictionary*.

each dictionary key specifies a Mailbox name;

if the authAccountName user is not specified or if the specified user has the `Select` access right for this Mailbox, the key value contains a *dictionary* with Mailbox information;

if the specified authAccountName does not have the `Select` access right, the key value contains an empty *array*;

if there is a 'mailbox folder' with the dictionary key, but there is no 'regular' Mailbox with that name, the key value is an empty *array*;

if there is a 'mailbox folder' with the dictionary key, and there is also a 'regular' Mailbox with that name, the key value is an *array* with one element - the information for the 'regular' Mailbox (either a dictionary or an empty array).

*accountName* : *string*

This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

*filter* : *string*

This optional parameter specifies the filter string to apply to Account Mailbox names. The filter can use the same wildcard symbols "*" and "%" as the IMAP LIST command. If the filter is not specified, the filter string "*" is assumed, and all Account Mailboxes are returned.

*authAccountName* : *string*

This optional parameter specifies the name of an Account on whose behalf the LIST operation should be executed. If this name is specified, the output includes only those Mailboxes for which the specified Account has the `Lookup` Mailbox access right.

CREATEMAILBOX *accountName* MAILBOX *mailboxName* [ CLASS *mailboxClass* ] [ AUTH *authAccountName* ]

Use this command to create a Mailbox in the specified Account.

*accountName* : *string*

This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

*mailboxName* : *string*

This parameter specifies the name for the new Mailbox.

*authAccountName* : *string*

This optional parameter specifies the name of an Account on whose behalf this operation should be executed.

*mailboxClass* : *string*

This optional parameter specifies the Mailbox class for the new Mailbox

DELETEMAILBOX *accountName* MAILBOX *mailboxName* [ AUTH *authAccountName* ]
DELETEMAILBOX *accountName* MAILBOXES *mailboxName* [ AUTH *authAccountName* ]

Use this command to remove a Mailbox from the specified Account. If the keyword MAILBOXES is used, all nested Mailboxes (submailboxes) are deleted, too.

*accountName* : *string*

This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

*mailboxName* : *string*

This parameter specifies the name of the Mailbox to be deleted.

*authaccountname* : *string*

This optional parameter specifies the name of an Account on whose behalf the operation should be executed. If this name is specified, the Mailbox is deleted only if the specified Account has the `Create` access right for the 'outer' Mailbox (this means that an Account should have the `Create` access right for the `Archive` Mailbox in order to delete the `Archive/March` Mailbox), and the specified Account should have the `DELETE` right for the specified Mailbox.

RENAMEMAILBOX *accountName* MAILBOX *mailboxName* INTO *newMailboxName* [ AUTH *authAccountName* ]
RENAMEMAILBOX *accountName* MAILBOXES *mailboxName* INTO *newMailboxName* [ AUTH *authAccountName* ]

Use this command to rename a Mailbox in the specified Account. If the keyword MAILBOXES is used, all nested Mailboxes (submailboxes) are renamed, too.

*accountName* : *string*

This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

*mailboxName* : *string*

This parameter specifies the name of the Mailbox to be renamed.

*newMailboxName* : *string*

This parameter specifies the new name for the Mailbox.

*authaccountname* : *string*

This optional parameter specifies the name of an Account on whose behalf the operation should be executed. If this name is specified, the Mailbox is renamed only if the specified Account has a right to perform the DELETEMAILBOX operation with the original Mailbox name and the CREATEMAILBOX operation with the new Mailbox name (see above).

GETMAILBOXINFO *accountName* MAILBOX *mailboxName* [ AUTH *authAccountName* ]

Use this command to get the internal information about the Account Mailbox. The command produces an output - a *dictionary* with the Mailbox internal information.

*accountName* : *string*

This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

*mailboxName* : *string*

This parameter specifies the name of an existing Mailbox in the specified Account.

*authaccountname* : *string*

This optional parameter specifies the name of an Account on whose behalf the operation should be executed. If this name is specified, the Mailbox info is returned only if the specified Account has the Select Mailbox access right.

GETMAILBOXACL *accountName* MAILBOX *mailboxName* [ AUTH *authAccountName* ]

Use this command to get the access control list for the Account Mailbox. The command produces an output - a *dictionary* with the Mailbox access elements.

*accountName* : *string*

This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

*mailboxName* : *string*

This parameter specifies the name of an existing Mailbox in the specified Account.

*authaccountname* : *string*

This optional parameter specifies the name of an Account on whose behalf the operation should be executed. If this name is specified, the ACL info is returned only if the specified Account has the Admin access right for the specified Mailbox.

SETMAILBOXACL *accountName* MAILBOX *mailboxName* [ AUTH *authAccountName* ] *newACL*

Use this command to modify the access control list for the Account Mailbox.

*accountName* : *string*

This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

843

*mailboxName* : *string*
        This parameter specifies the name of an existing Mailbox in the specified Account.

    *authaccountname* : *string*
        This optional parameter specifies the name of an Account on whose behalf the operation should be executed. If this name is specified, the ACL info is updated only if the specified Account has the `Admin` access right for the specified Mailbox.

    *newACL* : *dictionary*
        This parameter specifies the access right elements to be modified. Each dictionary key specifies an *identifier*, and the key value should be a string with access right symbols.
        If the key value string starts with the minus ("-") symbol, access rights specified in the string are removed from the access right element.
        If the key value string starts with the plus ("+") symbol, access rights specified in the string are added to the access right element.
        In other cases, access rights specified in the string replace the set of rights in the access right element.
        If the access right element for the specified key did not exist, it is created.
        If the new access right element has empty set of access rights, the element is removed.


GETMAILBOXRIGHTS *accountName* MAILBOX *mailboxName* AUTH *authAccountName*
    This command produces an output - a *string* with the effective Mailbox access rights for the given authAccountName.

    *accountName* : *string*
        This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

    *mailboxName* : *string*
        This parameter specifies the name of an existing Mailbox in the specified Account.

    *authaccountname* : *string*
        This parameter specifies the name of an Account whose effective access rights should be retrieved.


SETMAILBOXCLASS *accountName* MAILBOX *mailboxName* [ AUTH *authAccountName* ] CLASS *newClass*
    Use this command to set the "class" of an Account Mailbox.

    *accountName* : *string*
        This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

    *mailboxName* : *string*
        This parameter specifies the name of an existing Mailbox in the specified Account.

    *authaccountname* : *string*
        This optional parameter specifies the name of an Account whose Mailbox access rights should be used.

    *newClass* : *string*
        The Mailbox class.


GETMAILBOXSUBSCRIPTION *accountName*
    This command produces an output - an *array* with the list of Account "subscribed Mailboxes".

    *accountName* : *string*
        This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.


SETMAILBOXSUBSCRIPTION *accountName* *newSubscription*
    Use this command to set the Account "subscribed Mailboxes" list.

*accountName* : *string*
>> This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

*newSubscription* : *array*
>> The list of subscribed Mailboxes. Each array element should be a *string* with a Mailbox name.


GETMAILBOXALIASES *accountName*
> This command produces an output - a *dictionary*. Each dictionary key is the name of an existing Mailbox alias, and the key value is a *string* with the name of Mailbox this alias points to.

*accountName* : *string*
>> This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.


SETMAILBOXALIASES *accountName* *newAliases*
> Use this command to set the Account Mailbox aliases.

*accountName* : *string*
>> This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

*newAliases* : *dictionary*
>> The set of new Mailbox aliases.


---

# Alert Administration

A user should have the [All Domains](#) Server access right or the `CanPostAlerts` [Domain Administration access right](#) to use the [Alert](#) Administration CLI commands.

GETDOMAINALERTS [ *domainName* ]
> Use this command to get the Domain Alerts. The command produces an output - a *dictionary* with the Domain alert strings and time stamps.

*domainName* : *string*
>> This optional parameter specifies the name of an existing Domain.


SETDOMAINALERTS [ *domainName* ] *newAlerts*
> Use this command to change the Domain alerts.

*domainName* : *string*
>> This optional parameter specifies the name of an existing Domain.

*newAlerts* : *dictionary*
>> This dictionary is used to replace the Domain alert dictionary. All old Domain alerts are removed.


POSTDOMAINALERT *domainName* ALERT *newAlert*
> Use this command to post a Domain-wide alert message.

*domainName* : *string*
>> This parameter specifies the name of an existing Domain.

*newAlert* : *string*
>> This string specifies the Alert text.

REMOVEDOMAINALERT *domainName* ALERT *timeStamp*
>    Use this command to remove a Domain-wide alert message.

>    *domainName* : *string*
>    >    This parameter specifies the name of an existing Domain.

>    *timeStamp* : *string*
>    >    This string specifies the time stamp of the Alert message to be removed.


GETACCOUNTALERTS *accountName*
>    Use this command to get the Account Alerts. The command produces an output - a *dictionary* with the Account alert strings and time stamps.

>    *accountName* : *string*
>    >    This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.


SETACCOUNTALERTS *accountName* *newAlerts*
>    Use this command to change the Account alerts.

>    *accountName* : *string*
>    >    This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

>    *newAlerts* : *dictionary*
>    >    This dictionary is used to replace the Account alert dictionary. All old Account alerts are removed.


POSTACCOUNTALERT *accountName* ALERT *newAlert*
>    Use this command to post an Account alert message.

>    *accountName* : *string*
>    >    This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

>    *newAlert* : *string*
>    >    This string specifies the Alert text.


REMOVEACCOUNTALERT *accountName* ALERT *timeStamp*
>    Use this command to remove an Account alert message.

>    *accountName* : *string*
>    >    This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

>    *timeStamp* : *string*
>    >    This string specifies the time stamp of the Alert message to be removed.


The following commands are available for the System Administrators only:

GETSERVERALERTS
>    Use this command to get the list of the server-wide Alerts. The command produces an output - a *dictionary* with the server alert strings and time stamps.

```
SETSERVERALERTS newAlerts
```
Use this command to change the server-wide Alerts.

> ```
> newAlerts : dictionary
> ```
> This dictionary is used to replace the server-wide Alert dictionary. All old server-wide alerts are removed.

```
POSTSERVERALERT newAlert
```
Use this command to post a server-wide Alert message.

> ```
> newAlert : string
> ```
> This string specifies the Alert text.

```
REMOVESERVERALERT timeStamp
```
Use this command to remove a server-wide Alert message.

> ```
> timeStamp : string
> ```
> This string specifies the time stamp of the Alert message to be removed.

```
GETCLUSTERALERTS
SETCLUSTERALERTS newAlerts
POSTCLUSTERALERT newAlert
REMOVECLUSTERALERT timeStamp
```
These commands are available in the Dynamic Cluster only.
Use these commands instead of the [GET|SET|POST|REMOVE]SERVERALERT[S] commands to work with the cluster-wide Alerts.

---

# File Storage Administration

The following commands allow an authenticated user to deal with files in the Account File Storage area. To access File Storage:

- authenticated user should be the Account owner, or
- the authenticated user should have the All Domains Server access right or the WebSite Domain Administration access right, or
- the authenticated user should be granted a File Access Right to the specified file or directory (only if AUTH parameter is not specified)

If a file name ends with the slash (/) symbol, it specifies a file directory name.

```
READSTORAGEFILE accountName FILE fileName [ OFFSET position ] [ SIZE sliceSize ] [ AUTH authAccountName ]
```
Use this command to retrieve a file from the Account File Storage. This command produces an output - a *array* of 3 elements. The first element is a *datablock* with the content of the specified file, the second element is a timestamp with the file modification date, and the third element is a number equal to the current file size.

> ```
> accountName : string
> ```
> This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

> ```
> fileName : string
> ```
> This parameter specifies the name of the File Storage file to be retrieved.

> ```
> position : number
> ```
> If this parameter is specified the File Storage file is read starting from the specified file position.

> ```
> sliceSize : number
> ```
> If this parameter is specified, no more than the specified number of file data bytes is returned.

> ```
> authAccountName : string
> ```
> This optional parameter specifies the name of an Account on whose behalf the operation should be executed.

WRITESTORAGEFILE *accountName* FILE *fileName* [ OFFSET *position* ] [ AUTH *authAccountName* ] DATA *fileData*
    Use this command to store a file in the Account File Storage.
    If a File Storage file with the specified name already exists, the old file is removed.
    If the *fileName* specifies a directory (it ends with the slash (/) symbol) the command creates a directory. In this case, the
    OFFSET *position* part must be absent, and the *fileData* parameter must be an empty datablock.

    *accountName* : *string*
        This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current
        authenticated Account.

    *fileName* : *string*
        This parameter specifies the name for the File Storage file.

    *position* : *offset*
        If this parameter is absent, or it exists and it is the zero number, the existing file (if any) is removed first, and a new
        file is created.
        If this parameter is a non-zero number, its value must be positive; the File Storage file is rewritten/extended starting
        from the specified file position. The file should already exist, and the specified position should not be larger than the
        current file size.
        If this option is BEG, then the file should already exist, the file is rewritten from the beginning, but its old data beyond
        the end of the fileData (if any) is not removed.
        If this option is END, then the fileData is appended to the end of the file. If the file does not exist, it is created.
        If this option is NEW, then the file must not exist, a new file is created and fileData is stored in it.

    *authAccountName* : *string*
        This optional parameter specifies the name of an Account on whose behalf the operation should be executed.

    *fileData* : *datablock*
        This parameter contains the file data.

RENAMESTORAGEFILE *accountName* FILE *oldFileName* INTO *newFileName* [ AUTH *authAccountName* ]
    Use this command to rename a file or a file directory in the Account File Storage.

    *accountName* : *string*
        This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current
        authenticated Account.

    *oldFileName* : *string*
        This parameter specifies the name of an existing File Storage file or file directory.

    *newFileName* : *string*
        This parameter specifies the new name for the File Storage file or file directory.

    *authAccountName* : *string*
        This optional parameter specifies the name of an Account on whose behalf the operation should be executed.

DELETESTORAGEFILE *accountName* FILE *fileName* [ AUTH *authAccountName* ]
    Use this command to remove a file or a file directory from the Account File Storage.

    *accountName* : *string*
        This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current
        authenticated Account.

    *fileName* : *string*
        This parameter specifies the name of an existing File Storage file or file directory.

    *authAccountName* : *string*
        This optional parameter specifies the name of an Account on whose behalf the operation should be executed.

LISTSTORAGEFILES *accountName* [ PATH *filePath* ] [ AUTH *authAccountName* ]

    Use this command to list all files in the File Storage top directory or in one of its subdirectories. This command produces an output - a *dictionary*, where each key is a name of the File Storage file, and the key value is a *dictionary* for a regular file and an empty *array* for subdirectories.

    *accountName* : *string*
        This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

    *filePath* : *string*
        This optional parameter specifies the name of the File Storage subdirectory. You can omit this parameter along with the PATH keyword, in this case the command returns the list of files in the top File Storage directory.

    *authAccountName* : *string*
        This optional parameter specifies the name of an Account on whose behalf the operation should be executed.


GETSTORAGEFILEINFO *accountName* [ PATH *filePath* ] [ AUTH *authAccountName* ]

    Use this command to get the statistical information about all files in the Account File Storage. This command produces an output - an *array* with 2 *number* elements. The first element contains the total size of all File Storage files, the second element contains the number of files in the File Storage.

    *accountName* : *string*
        This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

    *authAccountName* : *string*
        This optional parameter specifies the name of an Account on whose behalf the operation should be executed.


READSTORAGEFILEATTR *accountName* FILE *fileName* [ *attributes* ] [ AUTH *authAccountName* ]

    Use this command to read attributes of an Account File Storage file or file directory. This command produces an output - an *array* of XML elements containing file or file directory attributes.

    *accountName* : *string*
        This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

    *fileName* : *string*
        This parameter specifies the name of an existing File Storage file or file directory.

    *attributes* : *array*
        This optional parameter specifies an array of strings. If specified, only file attributes with names included into this array are retrieved.

    *authAccountName* : *string*
        This optional parameter specifies the name of an Account on whose behalf the operation should be executed.


UPDATESTORAGEFILEATTR *accountName* FILE *fileName* *attributes* [ AUTH *authAccountName* ]

    Use this command to update attributes of an Account File Storage file or file directory.

    *accountName* : *string*
        This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

    *fileName* : *string*
        This parameter specifies the name of an existing File Storage file or file directory.

    *attributes* : *array*
        This parameter specifies an array of XML elements - the new file attribute values.

    *authAccountName* : *string*

This optional parameter specifies the name of an Account on whose behalf the operation should be executed.


GETFILESUBSCRIPTION *accountName*
> This command produces an output - an *array* with the list of Account "subscribed files".
>
> *accountName* : *string*
>> This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.


SETFILESUBSCRIPTION *accountName newSubscription*
> Use this command to set the Account "subscribed files" list.
>
> *accountName* : *string*
>> This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.
>
> *newSubscription* : *array*
>> The list of subscribed files. Each array element should be a *string* with a file name.


# Mailing Lists Administration

A user should have the [All Domains](#) Server access right or the [Domain Administrator](#) access right to use the Mailing List Administration CLI commands.

LISTLISTS [ *domainName* ]
> Use this command to get the list of all mailing lists in the Domain. The command produces output data - an *array* of strings. Each string is the name of a mailing list in the specified (or default) Domain.
>
> *domainName* : *string*
>> This optional parameter specifies the Domain name.


GETDOMAINLISTS [ *domainName* ]
> Use this command to get the list of all mailing lists in the Domain. The command produces output data - a *dictionary*. Each dictionary key is the name of a mailing list in the specified (or default) Domain. The key value is a numeric string with the actual number of the list subscribers ("-1" if the current number of subscribers is not known).
>
> *domainName* : *string*
>> This optional parameter specifies the Domain name.


GETACCOUNTLISTS *accountName*
> Use this command to get the list of all mailing lists belonging to the specified Account. The command produces output data - a *dictionary*. Each dictionary key is the name of a mailing list belonging to the specified (or default) Account. The key value is a numeric string with the actual number of the list subscribers ("-1" if the current number of subscribers is not known).
>
> *accountName* : *string*
>> This parameter specifies the list's owner Account name.


CREATELIST *listName* for *accountName*
> Use this command to create a mailing list.

*listName* : *string*
> This parameter specifies the name of a mailing list to create. It can include the Domain name. If the Domain name is not specified, the user Domain is used by default.

*accountName* : *string*
> This parameter specifies the name of the mailing list owner (without the Domain name). It should be the name of an already existing Account in the mailing list Domain.

Domain Administrators can use this command if they have the CanCreateLists Domain access right.

RENAMELIST *listName* into *newName*
> Use this command to rename a mailing list.

*listName* : *string*
> This parameter specifies the name of an existing mailing list. It can include the Domain name. If the Domain name is not specified, the user Domain is used by default.

*newName* : *string*
> This parameter specifies the new name for the mailing list (without the Domain part).

Domain Administrators can use this command if they have the CanCreateLists Domain access right.

DELETELIST *listName*
> Use this command to remove a mailing list.

*listName* : *string*
> This parameter specifies the name of an existing mailing list. It can include the Domain name. If the Domain name is not specified, the user Domain is used by default.

Domain Administrators can use this command if they have the CanCreateLists Domain access right.

The following commands can be used by the mailing list owner, by a Domain Administrator with the CanAccessLists access right, or by a Server Administrator with the [All Domains](#) Server access right.

GETLIST *listName*
> Use this command to retrieve list settings. The command produces an output - a *dictionary* with the listName mailing list settings.

*listName* : *string*
> This parameter specifies the name of an existing mailing list. It can include the Domain name. If the Domain name is not specified, the user Domain is used by default.

UPDATELIST *listName* *newSettings*
> Use this command to modify list settings.

*listName* : *string*
> This parameter specifies the name of an existing mailing list. It can include the Domain name. If the Domain name is not specified, the user Domain is used by default.

*newSettings* : *dictionary*
> This dictionary is used to update the mailing list settings dictionary. It does not have to contain all settings data, the omitted settings will be left unmodified.

LIST *listName* *operation* [silently] [confirm] *subscriber*
> Use this command to update the subscribers list.

*listName* : *string*
> This parameter specifies the name of an existing mailing list. It can include the Domain name. If the Domain name is

not specified, the user Domain is used by default.

*operation* : subscribe | feed | digest | index | null | banned | unsubscribe
This parameter specifies the operation (see the [LIST module](#) section for the details).

silently
This optional parameter tells the server not to send the Welcome/Bye message to the subscriber.

confirm
This optional parameter tells the server to send a confirmation request to the subscriber.

*subscriber* : *E-mail address*
The subscriber address. It can include the *comment part* used as the subscriber's real name.

Sample:
LIST MyList@mydomain.com FEED confirm "Bill Jones" <BJones@company.com>

LISTSUBSCRIBERS *listName* [ FILTER *filter* [ *limit* ] ]
Use this command to retrieve list subscribers. The command produces an output - an *array* with subscribers' E-mail addresses.

*listName* : *string*
This parameter specifies the name of an existing mailing list. It can include the Domain name. If the Domain name is not specified, the user Domain is used by default.

*filter* : *string*
If this optional parameter is specified, only the addresses containing the specified string are returned.

*limit* : *number*
This optional parameter limits the number of subscriber addresses returned.

READSUBSCRIBERS *listName* [ FILTER *filter* [ *limit* ] ]
Use this command to retrieve list subscribers. The command produces an output - an *array*, where the first element is a number - the total number of list subscribers, and the second element is an array of subscriber descriptor dictionaries.

*listName* : *string*
This parameter specifies the name of an existing mailing list. It can include the Domain name. If the Domain name is not specified, the user Domain is used by default.

*filter* : *string*
If this optional parameter is specified, only subscribers with addresses containing the specified string are returned.

*limit* : *number*
This optional parameter limits the number of subscriber dictionaries returned.

A dictionary describing a subscriber has the following elements:

Sub
E-mail address string

RealName
an optional string with Real name

mode
a string with subscription mode (index, digest, null, etc.)

subscribeTime
timestamp data specifying the moment when this user subscribed.

posts
number of postings on this list

lastBounceTime
optional timestamp data specifying the last time when messages sent to this user failed.

bounces
    optional numeric data specifying the number of failed delivery reports received for this user.

GETSUBSCRIBERINFO *listName* NAME *subscriberAddress*
    Use this command to retrieve information about a list subscriber. The command produces an output - a *dictionary* with subscriber information.

*listName* : *string*
    This parameter specifies the name of an existing mailing list. It can include the Domain name. If the Domain name is not specified, the user Domain is used by default.

*subscriberAddress* : *string*
    This parameter specifies the E-mail address of the list subscriber.

If the subscriber does not exist, an empty dictionary is returned. Otherwise, the dictionary contains the following elements:

*mode*
    This string element specified the subscription mode (digest, index, etc.) This element is equal to unsubscribe if the address has been unsubscribed, but has not been removed from the list. This element is equal to subscribe if a user has started subscription, but the subscription has not been confirmed.

*confirmationID*
    This element contains the subscriber's Confirmation ID string.

*timeSubscribed*
    This string element specifies when the address was subscribed (in the ACAP date/time format).

*posts*
    This string element may contain the strings special, moderateAll, prohibited, or the string with the number of messages posted from this address. If the next postings from this address are to be moderated, the element contains an array with one string element that contains the number of postings to be moderated.

*bounces*
    This optional string element contains the number of bounces received from this address.

*lastBounced*
    This optional string element specifies the last time when messages to this address bounced were bounced. The data and time are specified in the ACAP format.

*RealName*
    This optional string element contains the real name of the subscriber.

SETPOSTINGMODE *listName* FOR *subscriberAddress* [ UNMODERATED | MODERATEALL | PROHIBITED | SPECIAL | *numberOfModerated* ]
    Use this command to set the posting mode for the specified subscriber.

*listName* : *string*
    This parameter specifies the name of an existing mailing list. It can include the Domain name. If the Domain name is not specified, the user Domain is used by default.

*subscriberAddress* : *string*
    This parameter specifies the E-mail address of the list subscriber.

*postingMode* : *number*
    This optional parameter limits the number of subscriber addresses returned.

The command sets the posting mode the specified subscriber. If *numberOfModerated* (a number) is specified, the posting mode set requires moderation of the first numberOfModerated messages from this subscriber.

PROCESSBOUNCE *listName* [ FATAL ] FOR *subscriberAddress*

Use this command to perform the same action the List Manager performs when it receives a bounce message for the subscriber address.

*listName* : *string*
> This parameter specifies the name of an existing mailing list. It can include the Domain name. If the Domain name is not specified, the user Domain is used by default.

*subscriberAddress* : *string*
> This parameter specifies the E-mail address of the list subscriber.

Use the `FATAL` keyword to emulate a "fatal" bounce. Otherwise the command emulates a non-fatal bounce.

---

# Web Skins Administration

The following commands can be used to manage CommuniGate Pro [Skins](#) used for the CommuniGate Pro WebUser Interface.

A user should have the [All Domains](#) Server access right or the `CanModifySkins` [Domain Administration access right](#) to modify the Domain Skins.

LISTDOMAINSKINS [ *domainName* ]
> Use this command to list custom Domain Skins. The command produces an output - an *array* with Skin names.

*domainName* : *string*
> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain.

CREATEDOMAINSKIN [ *domainName* SKIN ] *skinName*
> Use this command to create a custom Domain Skin.

*domainName* : *string*
> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain. If it is specified, it should be followed with the SKIN keyword.

*skinName* : *string*
> This parameter specifies the name of the new Skin.

To create the unnamed Domain Skin, specify an empty string as the *skinName* parameter value.
A named Domain Skin can be created only when the unnamed Domain Skin exists.

RENAMEDOMAINSKIN [ *domainName* SKIN ] *skinName* INTO *newSkinName*
> Use this command to rename a custom named Domain Skin.

*domainName* : *string*
> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain. If it is specified, it should be followed with the SKIN keyword.

*skinName* : *string*
> This parameter specifies the name of an existing named Skin.

*newSkinName* : *string*
> This parameter specifies the new name for the Skin.

DELETEDOMAINSKIN [ *domainName* SKIN ] *skinName*
> Use this command to delete a custom Domain Skin.

*domainName* : *string*
> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the

administrator Domain. If it is specified, it should be followed with the SKIN keyword.

> *skinName* : *string*
>> This parameter specifies the name of the Skin to be deleted.

To delete the unnamed Domain Skin, specify an empty string as the *skinName* parameter value.
The unnamed Domain Skin can be deleted only when no named Domain Skin exists.

LISTDOMAINSKINFILES [ *domainName* SKIN ] *skinName*
> Use this command to list files in a custom Domain Skin. The command produces an output - a *dictionary* with Skin file names as keys. The dictionary element values are dictionaries with file attributes.

> *domainName* : *string*
>> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain. If it is specified, it must be followed with the SKIN keyword.

> *skinName* : *string*
>> This parameter specifies the name of an existing Domain Skin.

READDOMAINSKINFILE [ *domainName* SKIN ] *skinName* FILE *fileName*
> Use this command to read a file from a custom Domain Skin. The command produces an output - an *array*. The first array element is a datablock with the Skin file content, the second array element is a timestamp with the file modification date.

> *domainName* : *string*
>> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain. If it is specified, it must be followed with the SKIN keyword.

> *skinName* : *string*
>> This parameter specifies the name of an existing Domain Skin.

> *fileName* : *string*
>> This parameter specifies the name of an existing file in the specified Domain Skin.

STOREDOMAINSKINFILE [ *domainName* SKIN ] *skinName* FILE *fileName* DATA *fileContent*
STOREDOMAINSKINFILE [ *domainName* SKIN ] *skinName* FILE *fileName* DELETE
> Use this command to store a file into a custom Domain Skin, or to delete a file from a custom Domain Skin.

> *domainName* : *string*
>> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain. If it is specified, it must be followed with the SKIN keyword.

> *skinName* : *string*
>> This parameter specifies the name of an existing Domain Skin.

> *fileName* : *string*
>> This parameter specifies the Skin file name.

> *fileContent* : *datablock*
>> This datablock contains file content. This parameter is specified only if the DATA keyword is used.

If the DATA keyword is specified and the Skin contains a file with the same name, the old file is deleted. The file with the specified name is removed from the Skin Cache (in the Dynamic Cluster the file is removed from Skin caches on all cluster members).

The following commands are available for the System Administrators only:

LISTSERVERSKINS
> Use this command to list custom Server Skins. The command produces an output - an *array* with Skin names.

```
CREATESERVERSKIN skinName
```
Use this command to create a custom Server Skin.

> ```
> skinName : string
> ```
> This parameter specifies the name of the new Skin.

```
RENAMESERVERSKIN skinName INTO newSkinName
```
Use this command to rename a custom Server Skin.

> ```
> skinName : string
> ```
> This parameter specifies the name of an existing Skin.
>
> ```
> newSkinName : string
> ```
> This parameter specifies the new name for the Skin.

```
DELETESERVERSKIN skinName
```
Use this command to delete a custom Server Skin.

> ```
> skinName : string
> ```
> This parameter specifies the name of the Skin to be deleted.

```
LISTSERVERSKINFILES skinName
```
Use this command to list files in a custom Server Skin. The command produces an output - a *dictionary* with Skin file names as keys. The dictionary element values are dictionaries with file attributes.

> ```
> skinName : string
> ```
> This parameter specifies the name of an existing Server Skin.

```
READSERVERSKINFILE skinName FILE fileName
```
Use this command to read a file from a custom Server Skin. The command produces an output - an *array*. The first array element is a datablock with the Skin file content, the second array element is a timestamp with the file modification date.

> ```
> skinName : string
> ```
> This parameter specifies the name of an existing Server Skin.
>
> ```
> fileName : string
> ```
> This parameter specifies the name of an existing file in the specified Server Skin.

```
STORESERVERSKINFILE skinName FILE fileName DATA fileContent
STORESERVERSKINFILE skinName FILE fileName DELETE
```
Use this command to store a file into a custom Server Skin, or to delete a file from a custom Server Skin.

> ```
> skinName : string
> ```
> This parameter specifies the name of an existing Server Skin.
>
> ```
> fileName : string
> ```
> This parameter specifies the Skin file name.
>
> ```
> fileContent : datablock
> ```
> This datablock contains the file content. This parameter is specified only if the DATA keyword is used.

If the DATA keyword is specified and the Skin contains a file with the same name, the old file is deleted. The file with the specified name is removed from the Skin Cache (in the Dynamic Cluster the file is removed from Skin caches on all cluster members).

```
LISTCLUSTERSKINS
```

```
CREATECLUSTERSKIN skinName
RENAMECLUSTERSKIN skinName INTO newSkinName
DELETECLUSTERSKIN skinName
```
These commands are available in the Dynamic Cluster only.
Use these commands instead of the [LIST|CREATE|RENAME|DELETE]SERVERSKIN[S] commands to work with the cluster-wide Skins.

```
LISTCLUSTERSKINFILES skinName
READCLUSTERSKINFILE skinName FILE fileName
STORECLUSTERSKINFILE skinName FILE fileName DATA fileContent
STORECLUSTERSKINFILE skinName FILE fileName DELETE
```
These commands are available in the Dynamic Cluster only.
Use these commands instead of the [LIST|READ|STORE]SERVERSKINFILE[S] commands to work with files in the cluster-wide Skins.

```
LISTSTOCKSKINFILES skinName
READSTOCKSKINFILE skinName FILE fileName
```
Use these commands instead of the [LIST|READ]SERVERSKINFILE[S] commands to work with files in the built-in Skins.

---

# Web Interface Integration

The following commands can be used to integrate the CommuniGate Pro WebUser Interface with third-party applications.

```
CREATEWEBUSERSESSION accountName ADDRESS ip-address [ FOR orig-address ] [ WML | IMode ] [ SKIN skinName ]
```
Use this command to create a WebUser session for the specified Account. The command produces an output - a *string* that contains the WebUser Session ID. This string can be used to compose a URL that will allow the client browser to "enter" the WebUser Session. That URL can have the following format:
`http://cgateproserver:port/Session/rrrrrrrrrrr/Mailboxes.wssp`
where rrrrrrrrrrrr is the Session ID string returned.

> `accountName : string`
> This parameter specifies the Account name.

> `ip-address : string or IP address`
> This parameter specifies the IP address and port of the client browser.
> If the Account has the "Fixed IP" Preference setting enabled, connections to the session will be allowed from this IP address only.

> `orig-address : string`
> This parameter specifies the original IP address of the client browser, if the client connects via a proxy. The *ip-address* parameter specifies the proxy IP address.
> If the Account has the "Fixed IP" Preference setting enabled, connections to the session will be allowed from the proxy IP address only and only from this original IP address (passed by the proxy in the X-FORWARDED-FOR HTTP header field).

> `skinName : string`
> This optional parameter specifies the Skin to use for the newly created session.

The optional WML or IMode keywords can be used to emulate login via a WML or I-Mode browser.
The authenticated user should have the All Domains Server access right or the CanCreateWebUserSessions Domain Administration access right to create WebUser Sessions.

```
CREATEXIMSSSESSION accountName ADDRESS ip-address [ FOR orig-address ]
```
Use this command to create a XIMSS session for the specified Account. The command produces an output - a *string* that contains the XIMSS Session ID. This string can be used to compose a URL that will allow the client browser to work with the XIMSS Session using HTTP Binding.

> `accountName : string`
> This parameter specifies the Account name.

```
ip-address : string
orig-address : string
```
These parameters have the same meaning as for the CREATEWEBUSERSESSION command.

The authenticated user should have the [All Domains](#) Server access right or the CanCreateWebUserSessions [Domain Administration](#) access right to create XIMSS Sessions.

FINDACCOUNTSSESSION *accountName* [ ADDRESS *ip-address* [ FOR *proxied-address* ] ] [ PROTOCOL *protocol* ] [ TRANSPORT *transport* ] [ CLIENT *client* ]
Use this command to find an existing session for the specified Account. The command produces an output - a *string* that contains the Session ID.

```
accountName : string
```
This parameter specifies the Account name.

```
ip-address : string or IP address
```
This optional parameter specifies the IP address of the client browser. If it is specified, the command will find only those sessions that have the "Fixed IP" Preference disabled or have the same login IP address as the specified one.

```
proxied-address : string
```
This optional parameter specifies the IP address of the client browser, if this browser is located behind an HTTP proxy. The *ip-address* then specifies the IP address of that proxy.

```
protocol : string
```
This optional parameter specifies the Session protocol (WebUser, XIMSS, XMPP, etc.) If specified, only the sessions created with the specified protocol are searched.

```
transport : string
```
This optional parameter specifies the Session transport (HTTP, XIMSS, XMPP, etc.) If specified, only the sessions created with the specified transport are searched.

```
client : string
```
This optional parameter specifies the Session client. If specified, only the sessions created with the specified client (if the client has informed the session about its name) are searched.

The authenticated user should have the [All Domains](#) Server access right or the CanCreateWebUserSessions [Domain Administration](#) access right to use this command.

LISTACCOUNTSSESSIONS *accountName* [ ADDRESS *ip-address* [ FOR *proxied-address* ] ] [ PROTOCOL *protocol* ] [ TRANSPORT *transport* ] [ CLIENT *client* ]
Use this command to retrieve all existing sessions for the specified Account. The command produces an output - an *array* of strings, where each string is the Session ID.
Command parameters are the same as the FINDACCOUNTSSESSION command parameters.
The authenticated user should have the [All Domains](#) Server access right or the CanCreateWebUserSessions [Domain Administration](#) access right to use this command.

GETSESSION *sessionID* [ DOMAIN *domainName* ]
Use this command to retrieve Session data. The command produces an output - a *dictionary* with the *session dataset* (specified in the [WSSP](#) section of this manual).

```
sessionID : string
```
This parameter specifies the Session ID.

```
domainName : string
```
This optional parameter specifies the name of Domain the session Account belongs to.

The authenticated user should have the [All Domains](#) Server access right to retrieve Session data if the *domainName* parameter is not specified. If the *domainName* is specified, the authenticated user should have the CanCreateWebUserSessions [Domain Administration](#) access right for the specified Domain.
This operation resets the session inactivity timer.

KILLSESSION *sessionID* [ DOMAIN *domainName* ]

Use this command to terminate a Session.

    *sessionID* : *string*
        This parameter specifies the Session ID.

    *domainName* : *string*
        This optional parameter specifies the name of Domain the session Account belongs to.


The authenticated user should have the [All Domains](#) Server access right to terminate a Session if the *domainName* parameter is not specified. If the *domainName* is specified, the authenticated user should have the `CanCreateWebUserSessions` [Domain Administration](#) access right for the specified Domain.

BLESSSESSION *sessionID* [ PASSWORD *secret* ] [ DOMAIN *domainName* ]
    Use this command to complete the second stage of a Two-factor authentication process for the given session.

    *sessionID* : *string*
        This parameter specifies the Session ID.

    *secret* : *string*
        This optional parameter specifies the one-time secret used with Two-factor Authentication.

    *domainName* : *string*
        This optional parameter specifies the name of Domain the session Account belongs to.


The authenticated user should have the [All Domains](#) Server access right to complete the Two-factor Authentication process for a Session if the *domainName* parameter is not specified. If the *domainName* is specified, the authenticated user should have the `CanCreateWebUserSessions` [Domain Administration](#) access right for the specified Domain. If the *secret* parameter is not specified then the Session should be waiting the Two-factor Authentication process completion in background and authenticated user should have the `Master` [Server Administration](#) access right.

---

# Real-Time Application Administration

The following commands can be used to manage CommuniGate Pro [Real-Time](#) Application Environments.

A user should have the [All Domains](#) Server access right or the `CanModifyPBXApps` [Domain Administration access right](#) to modify the Domain Real-Time Application Environment.

CREATEDOMAINPBX domainName [ FILE *language* ]
    Use this command to create the Domain Real-Time Application Environment or to create its national subset.

    *domainName* : *string*
        This parameter specifies the Domain name.

    *language* : *string*
        This optional parameter specifies a national subset name.


DELETEDOMAINPBX domainName FILE *language*
    Use this command to remove a national subset from the Domain Real-Time Application Environment.

    *domainName* : *string*
        This parameter specifies the Domain name.

    *language* : *string*
        This parameter specifies a national subset name.


LISTDOMAINPBXFILES domainName [ FILE *language* ]
    Use this command to list files in the Domain Real-Time Application Environment. The command produces an output - a

*dictionary* with file names used as keys. The dictionary element values are dictionaries with file attributes.

> `domainName` : *string*
>> This optional parameter specifies the Domain name. If the Domain name is not specified, the command applies to the administrator Domain.

> `language` : *string*
>> This optional parameter specifies a national subset name.

`READDOMAINPBXFILE domainName FILE` *fileName*
> Use this command to read a file from the Domain Real-Time Application Environment. The command produces an output - a *datablock* with the file contents.

> `domainName` : *string*
>> This parameter specifies the Domain name.

> `fileName` : *string*
>> This parameter specifies the file name. To retrieve a file from a national subset, specify the name as *language/fileName*.

`STOREDOMAINPBXFILE domainName FILE` *fileName* `DATA` *fileContent*
`STOREDOMAINPBXFILE domainName FILE` *fileName* `DELETE`
> Use this command to store a file into the Domain Real-Time Application Environment, or to delete a file from the Domain Real-Time Application Environment.

> `domainName` : *string*
>> This parameter specifies the Domain name.

> `fileName` : *string*
>> This parameter specifies the file name. To store a file into a national subset, specify the name as *language/fileName*.

> `fileContent` : *datablock*
>> This parameter is specified only if the DATA keyword is used. It should contain the file contents.

> If the DATA keyword is specified and the environment contains a file with the specified name, the old file is deleted. The file with the specified name is removed from the Environment cache (in the Dynamic Cluster the file is removed from all cluster members caches).

The following commands are available for the System Administrators only:

`CREATESERVERPBX` *language*
> Use this command to create the Server-wide Real-Time Application Environment or to create its national subset.

> `language` : *string*
>> This parameter specifies a national subset name.

`DELETESERVERPBX` *language*
> Use this command to remove a national subset of the Server-wide Real-Time Application Environment.

> `language` : *string*
>> This parameter specifies a national subset name.

`LISTSERVERPBXFILES [` *language* `]`
> Use this command to list files in the Server-wide Real-Time Application Environment. The command produces an output - a *dictionary* with file names used as keys. The dictionary element values are dictionaries with file attributes.

> `language` : *string*

This optional parameter specifies a national subset name.

**READSERVERPBXFILE** *fileName*

Use this command to read a file from the Server-wide Real-Time Application Environment. The command produces an output - a *datablock* with the file contents.

> *fileName* : *string*
>
> > This parameter specifies the file name. To retrieve a file from a national subset, specify the name as *language*/*fileName*.

**STORESERVERPBXFILE** *fileName* **DATA** *fileContent*
**STORESERVERPBXFILE** *fileName* **DELETE**

Use this command to store a file into the Server-wide Real-Time Application Environment, or to delete a file from the Server-wide Real-Time Application Environment.

> *fileName* : *string*
>
> > This parameter specifies the file name. To store a file into a national subset, specify the name as *language*/*fileName*.
>
> *fileContent* : *datablock*
>
> > This parameter is specified only if the DATA keyword is used. It should contain the file contents.

If the DATA keyword is specified and the environment contains a file with the specified name, the old file is deleted. The file with the specified name is removed from the Environment cache (in the Dynamic Cluster the file is removed from all cluster members caches).

**CREATECLUSTERPBX** *language*

**DELETECLUSTERPBX** *language*

**LISTCLUSTERPBXFILES** [ *language* ]

**READCLUSTERPBXFILE** *fileName*

**STORECLUSTERPBXFILE** *fileName* **DATA** *fileContent*
**STORECLUSTERPBXFILE** *fileName* **DELETE**

These commands are available in the Dynamic Cluster only.
Use these commands instead of the [LIST|READ|STORE]SERVERPBXFILE[S] commands to work with files in the cluster-wide Real-Time Application Environment.

**LISTSTOCKPBXFILES** [ *language* ]
**READSTOCKPBXFILE** *fileName*

Use these commands instead of the [LIST|READ]SERVERPBXFILE[S] commands to work with files in the stock (built-in) Real-Time Application Environment.

---

# Real-Time Application Control

The following commands can be used to manage CommuniGate Pro [Real-Time](#) Application Tasks.

**STARTPBXTASK** *accountName* **PROGRAM** *programName* [ **ENTRY** *entryName* ] [ **PARAM** *parameter* ]

Use this command to start a new PBX Task. The command produces an output - a *string* with the Task ID.

> *accountName* : *string*
>
> > This parameter specifies the name of an Account. The Task is started on this Account behalf.
> > The name can include the Domain name. If the Domain name is not specified, the current user Domain is used by default.
>
> *programName* : *string*
>
> > The name of the program (the .sppr file) to start.
>
> *entryName* : *string*
>
> > This optional parameter specifies the program entry point. If this parameter is not specified, the `main` entry point is

used.

> *parameter* : *object*
>> This optional parameter specifies the program parameter. The program code can retrieve it using the following code:
>> `Vars().startParameter`

---

`SENDTASKEVENT` *taskID* `EVENT` *eventName* [ `PARAM` *parameter* ]
> Use this command to send an Event to an existing PBX Task.

> *taskID* : *string*
>> This parameter specifies the Task ID.

> *eventName* : *string*
>> The name of the Event to send.

> *parameter* : *object*
>> This optional parameter specifies the Event parameter.

---

`KILLNODE` *taskID*
> Use this command to kill an existing PBX Task.

> *taskID* : *string*
>> This parameter specifies the Task ID.

---

`READNODESTATUS` *taskID*
> Use this command to read the current [application status](#) of an existing PBX Task. The command produces an output - the application status object.

> *taskID* : *string*
>> This parameter specifies the Task ID.

---

# Account Services

The following commands can be used to manage various CommuniGate Pro Account Services.

`REMOVEACCOUNTSUBSET` *accountName* `SUBSET` *subsetName*
> Use this command to remove an Account "dataset" (such as the `RepliedAddresses` dataset).

> A user should be the Account owner or should have the [BasicSettings](#) Domain Administration access right to use this command.

> *accountName* : *string*
>> This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

> *subsetName* : *string*
>> This parameter specifies the name of an existing data subset in the specified Account.

---

`DATASET` *accountName* *parameters*
> Use this command to manage Account "datasets". The command produces an output - a *dictionary* with the operation results.

> A user should be the Account owner or should have the [BasicSettings](#) Domain Administration access right to use this command.

*accountName* : *string*
>    This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

*parameters* : *dictionary*
>    This dictionary should contain:

>    subsetName
>    >    a string element specifying the target dataset or the dataset subset

>    what
>    >    a string element specifying the operation to apply.

>    Other dictionary elements are operation-specific.

The following is the list of supported operations (the `what` values) and the additional *parameters* dictionary elements used for each operation:

listSubsets
>    this operation lists all subsets of the specified dataset. To list all top-level datasets in the Account, specify the an empty string as the `subsetName` value. The resulting dictionary contains the found subset names as keys and empty strings as values.

createSet
>    this operation creates the specified dataset.

removeSet
>    this operation removes the specified dataset.

listEntries
>    this operation lists subset entries. The resulting dictionary contains the found entry names as keys and the entry attribute name-value dictionaries as values.

>    attribute, data
>    >    optional string elements; they specify the name and the value of an entry attribute. If specified, the result includes only the entries that have the specified attribute with the specified value.
>    >    Use the `entry` attribute name to filter by entry names.

>    mode
>    >    an optional string element; if it is absent or its value is `eq`, then the specified attribute should have the specified value;
>    >    if its value is `beg`, then the beginning of the specified attribute value should match the specified value.
>    >    if its value is `end`, then the tail of the specified attribute value should match the specified value.
>    >    if its value is `incl`, then the specified attribute value should include the specified value.

setEntry
>    this operation creates a new entry or updates an existing entry.

>    data
>    >    a dictionary with the attribute name-value pairs; they are used to update an existing entry or to create a new one.

>    entryName
>    >    the entry name string; if the entry with the specified name does not exist, it is created. If this element is absent, a unique entry name is generated.

>    ifExists
>    >    if this element exists, then the new entry cannot be created, and only an existing entry can be updated; if this element is absent and the specified dataset is not found, the dataset is created.

deleteEntry
> this operation removes the specified entry from the specified dataset.

> entryName
> > the entry name string

addRandomEntry
> this operation adds a new entry to the specified dataset or the dataset subset. A unique name is generated and assigned to this entry. If the operation succeeds, the resulting dictionary has the string `entryName` element with the entry name generated.

> data
> > a dictionary with the attribute name-value pairs. It must contain the addressbook.Email attribute.

> entryLimit
> > an optional numeric value; if specified and positive, then the operation checks the the current number of subset entries does not exceed this limit.

> If the dataset already contains an entry with the same addressbook.Email attribute value, the dataset is not modified.

findAddress
> this operation finds an entry with the specified addressbook.Email attribute value. The operation result is a dictionary. If an entry is found, its name is returned as the dictionary element with an empty-string name.

> address
> > a string with an E-mail address to look for

ROSTER *accountName parameters*
> Use this command to manage Account Roster. The command produces an output - a *dictionary* with the operation results.

> A user should be the Account owner or should have the [BasicSettings](#) Domain Administration access right to use this command.

> *accountName* : *string*
> > This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

> *parameters* : *dictionary*
> > This dictionary should contain the `what` string element, specifying the operation to apply: `List`, `Update`, `remove`, `Presence`, `probe`. Other dictionary elements are operation-specific.

BALANCE *accountName parameters*
> Use this command to manage Account Billing Balances. The command produces an output - a *dictionary* with the operation results (as specified in the [Billing](#) section).

> A user should be the Account owner or should have the [CanCreditAccounts](#) Domain Administration access right to use this command.

> *accountName* : *string*
> > This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

> *parameters* : *dictionary*
> > This dictionary should contain the `op` string element, specifying the operation to apply: `list`, `reserve`, `release`, `charge`, `credit`, `read`, `readAll`, `history`, `remove`. Other dictionary elements are operation-specific, they are specified in the [Billing](#) section.

# Server Settings

A user should have the [Settings](#) Server access right to use the Server Settings CLI commands.

`LISTMODULES`
> Use this command to list all Server modules. The command produces an output - an *array* with all module names.

`GETMODULE` *moduleName*
> Use this command to get the module settings. The command produces an output - a *dictionary* with the module settings.
>
> > *moduleName* : *string*
> > > This parameter specifies the name of a CommuniGate Pro Server module.

`SETMODULE` *moduleName newSettings*
> Use this command to set the module settings.
>
> > *moduleName* : *string*
> > > This parameter specifies the name of a CommuniGate Pro Server module.
> >
> > *newSettings* : *dictionary*
> > > This dictionary is used to set the module settings dictionary.

`UPDATEMODULE` *moduleName newSettings*
> Use this command to update the module settings.
>
> > *moduleName* : *string*
> > > This parameter specifies the name of a CommuniGate Pro Server module.
> >
> > *newSettings* : *dictionary*
> > > This dictionary is used to update the module settings dictionary. It does not have to contain all settings data, the omitted settings will be left unmodified.

`GETQUEUESETTINGS`
> Use this command to get the Queue settings. The command produces an output - a *dictionary* with the Queue settings.

`SETQUEUESETTINGS` *newSettings*
> Use this command to set the Queue settings.
>
> > *newSettings* : *dictionary*
> > > This dictionary is used to set the Queue settings dictionary.

`GETSIGNALSETTINGS`
> Use this command to get the Signal component settings. The command produces an output - a *dictionary* with the component settings.

`SETSIGNALSETTINGS` *newSettings*
> Use this command to set the Signal component settings.
>
> > *newSettings* : *dictionary*
> > > This dictionary is used to set the component settings dictionary.

`GETMEDIASERVERSETTINGS`

Use this command to read the Media Server component settings. The command produces an output - a *dictionary* with the component settings.

SETMEDIASERVERSETTINGS *newSettings*
     Use this command to set the Media Server component settings.

    *newSettings* : *dictionary*
        This dictionary is used to set the component settings dictionary.

GETSESSIONSETTINGS
     Use this command to get the user Sessions settings. The command produces an output - a *dictionary* with the Sessions settings.

SETSESSIONSETTINGS *newSettings*
     Use this command to set the user Sessions settings.

    *newSettings* : *dictionary*
        This dictionary is used to set the Sessions settings dictionary.

GETCLUSTERSETTINGS
     Use this command to get the Cluster settings. The command produces an output - a *dictionary* with the Cluster settings.

SETCLUSTERSETTINGS *newSettings*
     Use this command to set the Cluster settings.

    *newSettings* : *dictionary*
        This dictionary is used to set the Cluster settings dictionary.

GETLOGSETTINGS
     Use this command to get the Main Log settings. The command produces an output - a *dictionary* with the Main Log settings.

UPDATELOGSETTINGS *newSettings*
     Use this command to set the Main Log settings.

    *newSettings* : *dictionary*
        This dictionary is used to update the Main Log settings dictionary.

GETNETWORK
     Use this command to retrieve the Network settings. The command produces an output - a *dictionary* with the server Network settings.

SETNETWORK *newSettings*
     Use this command to set the server Network Settings.

    *newSettings* : *dictionary*
        New server Network settings.

GETDNRSETTINGS
     Use this command to retrieve the DNR (Domain Name Resolver) settings. The command produces an output - a *dictionary* with the DNR settings.

SETDNRSETTINGS *newSettings*
     Use this command to set the DNR (Domain Name Resolver) settings.

    *newSettings* : *dictionary*
        New DNR settings.

GETBANNED
    Use this command to retrieve the Banned Message Lines settings. The command produces an output - a *dictionary* with the
    server Banned Message Lines settings.

SETBANNED *newSettings*
    Use this command to set the server Banned Message Line Settings.

    *newSettings* : *dictionary*
        New server Banned settings.

GETCLUSTERNETWORK
SETCLUSTERNETWORK *newSettings*
    Use these commands to retrieve and update the Cluster-wide Network settings.

GETCLUSTERBANNED
SETCLUSTERBANNED *newSettings*
    Use these commands to retrieve and update the Cluster-wide Banned Message Lines settings.

GETSERVERMAILRULES
    Use this command to read the Server-Wide Automated Mail Processing Rules. The command produces an output - an *array*
    of the Server Queue Rules.

SETSERVERMAILRULES *newRules*
    Use this command to set the Server-Wide Automated Mail Processing Rules.

    *newRules* : *array*
        An array of new Server Queue Rules.

GETSERVERSIGNALRULES
    Use this command to read the Server-Wide Automated Signal Processing Rules. The command produces an output - an
    *array* of the Server Signal Rules.

SETSERVERSIGNALRULES *newRules*
    Use this command to set the Server-Wide Automated Signal Processing Rules.

    *newRules* : *array*
        An array of new Server Signal Rules.

GETCLUSTERMAILRULES
SETCLUSTERMAILRULES *newRules*
GETCLUSTERSIGNALRULES
SETCLUSTERSIGNALRULES *newRules*
    Use these commands to retrieve and update the Cluster-wide Rules.

GETROUTERTABLE
    Use this command to read the Router Table. The command produces an output - a (multi-line) *string* with the Router Table
    text.

SETROUTERTABLE *newTable*
    Use this command to set the Router Table.

    *newTable* : *string*
        A (multi-line) string containing the text of the new Router Table
        Note: multiple lines should be separated with the \e symbols.

GETROUTERSETTINGS
    Use this command to read the Router settings. The command produces an output - a *dictionary* with the Router settings.

SETROUTERSETTINGS *newSettings*
>    Use this command to set the Router settings.

>    *newSettings* : *dictionary*
>>        A dictionary containing new Router settings.

GETCLUSTERROUTERTABLE
SETCLUSTERROUTERTABLE *newTable*
GETCLUSTERROUTERSETTINGS
SETCLUSTERROUTERSETTINGS *newSettings*
>    Use these commands to deal with the Cluster-Wide Router Table and settings.

GETSERVERSETTINGS
>    Use this command to read the Server "other" settings. The command produces an output - a *dictionary* with the Server settings.

UPDATESERVERSETTINGS *newSettings*
>    Use this command to update the "other" Server settings.

>    *newSettings* : *dictionary*
>>        A dictionary containing new Server settings.

REFRESHOSDATA
>    Use this command to make the Server re-read the IP data from the server OS: the set of the local IP addresses, and the set of the DNS addresses.

GETLANIPS
>    Use this command to retrieve the set of LAN IP Addresses. The command produces an output - a (multi-line) *string* with LAN IP addresses and address ranges.

SETLANIPS *newAddresses*
>    Use this command to update the set of LAN IP Addresses.

>    *newAddresses* : *string*
>>        This (multi-line) string parameter contains the set of addresses and address ranges forming the new set of LAN IP Addresses.

GETCLUSTERLANIPS
>    Use this command to retrieve the set of Cluster-wide LAN IP Addresses. The command produces an output - a (multi-line) *string* with Cluster-wide LAN IP addresses and address ranges.

SETCLUSTERLANIPS *newAddresses*
>    Use this command to update the set of Cluster-wide LAN IP Addresses.

>    *newAddresses* : *string*
>>        This (multi-line) string parameter contains the set of addresses and address ranges forming the new set of Cluster-wide LAN IP Addresses.

The following command sets have the same parameters and outputs as the GETLANIPS | SETLANIPS | GETCLUSTERLANIPS | SETCLUSTERLANIPS command set:

GETCLIENTIPS
SETCLIENTIPS *newAddresses*
GETCLUSTERCLIENTIPS
SETCLUSTERCLIENTIPS *newAddresses*
>    Use these commands to retrieve and set Server-wide and Cluster-wide Client IP Addresses.

```
GETBLACKLISTEDIPS
SETBLACKLISTEDIPS newAddresses
GETCLUSTERBLACKLISTEDIPS
SETCLUSTERBLACKLISTEDIPS newAddresses
```
> Use these commands to retrieve and set Server-wide and Cluster-wide Blacklisted IP Addresses.

```
GETWHITEHOLEIPS
SETWHITEHOLEIPS newAddresses
GETCLUSTERWHITEHOLEIPS
SETCLUSTERWHITEHOLEIPS newAddresses
```
> Use these commands to retrieve and set Server-wide and Cluster-wide WhiteHole IP Addresses.

```
GETNATEDIPS
SETNATEDIPS newAddresses
GETCLUSTERNATEDIPS
SETCLUSTERNATEDIPS newAddresses
```
> Use these commands to retrieve and set Server-wide and Cluster-wide NATed IP Addresses.

```
GETNATSITEIPS
SETNATSITEIPS newAddresses
GETCLUSTERNATSITEIPS
SETCLUSTERNATSITEIPS newAddresses
```
> Use these commands to retrieve and set Server-wide and Cluster-wide NAT Site IP Addresses.

```
GETDEBUGIPS
SETDEBUGIPS newAddresses
GETCLUSTERDEBUGIPS
SETCLUSTERDEBUGIPS newAddresses
```
> Use these commands to retrieve and set Server-wide and Cluster-wide Debug IP Addresses.

```
GETDENIEDIPS
SETDENIEDIPS newAddresses
GETCLUSTERDENIEDIPS
SETCLUSTERDENIEDIPS newAddresses
```
> Use these commands to retrieve and set Server-wide and Cluster-wide Denied IP Addresses.

A user should have the Settings or User Server access right or the to use the following CLI commands.

```
ROUTE address [ mail | access | signal ]
```
> Use this command to get the routing for the specified address.

> ```
> address : string
> ```
> > This parameter specifies the E-mail address to be processed with the CommuniGate Pro Router.

> `mail` or `access` or `signal`
> > This optional flag specifies the Routing type (see the Router section for more details). The default mode is `access`.

> This command produces an output - an array of three strings:

> *module*
> > the name of the CommuniGate Pro module the address is routed to, or SYSTEM if the address is routed to a built-in destination (like NULL).

> *host*
> > the object/queue handled by the specified module: an Internet domain name for the SMTP module, a local Account name for the Local Delivery module, etc.

> *address*
> > the address inside the queue (E-mail address for SMTP, Real-To: address for Local Delivery, etc.)

```
GETIPSTATE ip-address [ TEMP ]
```
Use this command to get the type assigned to the specified address. The command produces an output - a *string* with the IP address type.
If the TEMP keyword is specified, the temporary Client IP Addresses set is checked.

> ```
> ip-address : string or IP address
> ```
> This parameter specifies the IP Address to check.

A user should have the [Master](#) Server access right to use the following CLI commands.

```
GETSERVERINTERCEPT
```
Use this command to read the Lawful Intercept settings. The command produces an output - a *dictionary* with the Intercept settings.

```
SETSERVERINTERCEPT newSettings
```
Use this command to set the Lawful Intercept settings.

> ```
> newSettings : dictionary
> ```
> A dictionary containing new Intercept settings.

```
GETCLUSTERINTERCEPT
SETCLUSTERINTERCEPT newSettings
```
These commands are the same as the GETSERVERINTERCEPT and SETSERVERINTERCEPT commands, but they deal with the Cluster-Wide Lawful Intercept settings.

---

# Monitoring

A user should have the [Monitoring](#) Server access right to use the Server Monitoring CLI commands.

```
GETSTATELEMENT ObjectID
```
Use this command to retrieve the current value of a Server statistics (SNMP) element.

> ```
> ObjectID : string
> ```
> The object ID of the Server statistics element (see the [Statistics](#) section for more details).

This command produces an output - a *number*, *string*, or other object with the Server statistics element value.

```
SETSTATELEMENT ObjectID [ INC | SET ] setValue
```
Use this command to update the current value of a Server statistics (SNMP) element. Only the "Custom" elements can be updated.

> ```
> ObjectID : string
> ```
> The object ID of the Server statistics element (see the [Statistics](#) section for more details).

> ```
> setValue : numeric string
> ```
> if the INC keyword is used, this value is added to the Element value, if the SET keyword is used, this value is assigned to the Element.

```
GETNEXTSTATNAME ObjectID
```
Use this command to enumerate available Server statistics (SNMP) elements.

> ```
> ObjectID : string
> ```
> An empty string or the object ID of the already found Server statistics element (see the [Statistics](#) section for more

details).

This command produces an output - a *string* with the ObjectID of the next statistics element.
If the ObjectID parameter is an empty string, the ObjectID of the first available Server statistics element is returned.
If a statistics element for the specified ObjectID is not found, or if the found element is the last available one, the command returns an error.

GETDIALOGINFO *DialogID*
> Use this command to retrieve the information about a Signal Dialog object.

> *DialogID* : *number*
>> The Dialog ID.

> This command produces an output - a *dictionary* with the Dialog status data.

SHUTDOWN
> Use this command to stop the CommuniGate Pro Server.

---

# Statistics

The Account-Level Statistics data is collected if the Account Statistics option is enabled.
To enable this option, open the General pages in the Settings realm of the CommuniGate Pro WebAdmin Interface, and find the Local Account Manager panel on the Other page.

GETACCOUNTSTAT *accountName* [ KEY *keyName* ]
> Use this command to retrieve statistics data about the specified Account.

> *accountName* : *string*
>> This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

> *keyName* : *string*
>> This optional parameter specifies the name of the statistical entry to retrieve.

> This command produces an output - a *number* or a *timeStamp* with the specified statistical information, or (if the KEY keyword and the *keyName* parameter are not specified) a *dictionary* with all available statistical data.
> If the statistical data for the specified key does not exist, an empty *string* is returned.
> To use this command, the user should have the Domain Administration right for the target Account Domain. All users can retrieve the Account statistics data for their own accounts.

RESETACCOUNTSTAT *accountName* [ KEY *keyName* ]
> Use this command to reset statistics data about the specified Account.

> *accountName* : *string*
>> This parameter specifies the name of an existing Account. The asterisk (*) symbol can be used to specify the current authenticated Account.

> *keyName* : *string*
>> This optional parameter specifies the name of the statistical entry to reset.

> If the KEY keyword and the *keyName* parameter are not specified, all Account statistical entries are reset.
> To use this command, the user should have the "Basic Settings" Domain Administration right for the target Account Domain.

The following Account statistics data keys are implemented:

| Key Name | Value |
| --- | --- |
| StatReset | The date & time when the last parameterless RESETACCOUNTSTAT command was sent to this Account |
| MessagesReceived | The total number of messages delivered to the Account |
| BytesReceived | The total size of all messages delivered to the Account |
| MessagesSent | The total number of messages sent on the Account behalf |
| BytesSent | The total size of all messages sent on the Account behalf |
| CallsReceived | The total number of calls received for the Account |
| CallsSent | The total number of calls placed on the Account behalf |
| Logins | The total number of successful Account authentications |

GETDOMAINSTAT *domainName* [ KEY *keyName* ]
>    Use this command to retrieve statistics data about the specified Domain.

>    *domainName* : *string*
>>        This parameter specifies the name of an existing Domain. The asterisk (*) symbol can be used to specify the Domain of the current authenticated Account.

>    *keyName* : *string*
>>        This optional parameter specifies the name of the statistical entry to retrieve.

>    This command produces an output - a *string* with the specified statistical information, or (if the KEY keyword and the *keyName* parameter are not specified) a *dictionary* with all available statistical data.
>    To use this command, the user should have the Domain Administration right for the target Domain.

RESETDOMAINSTAT *domainName* [ KEY *keyName* ]
>    Use this command to reset statistics data about the specified Domain.

>    *domainName* : *string*
>>        This parameter specifies the name of an existing Domain. The asterisk (*) symbol can be used to specify the Domain of the current authenticated Account.

>    *keyName* : *string*
>>        This optional parameter specifies the name of the statistical entry to reset.

>    If the KEY keyword and the *keyName* parameter are not specified, all Domain statistical entries are reset.
>    To use this command, the user should have the "Basic Settings" Domain Administration right for the target Domain.

The following Domain statistics data keys are implemented:

| Key Name | Value |
| --- | --- |
| StatReset | The date & time when the last parameterless RESETDOMAINSTAT command was sent to this Domains |
| MessagesReceived | The total number of messages delivered to the Domain Accounts |
| BytesReceived | The total size of all messages delivered to the Domain Accounts |
| MessagesSent | The total number of messages sent on the Domain Accounts behalf |
| BytesSent | The total size of all messages sent on the Domain Accounts behalf |
| CallsReceived | The total number of calls received by the Domain Accounts |
| CallsSent | The total number of calls placed on the Domain Accounts behalf |

# Directory Administration

A user should have the [Directory](#) Server access right to use the Directory Administration CLI commands.

```
LISTDIRECTORYUNITS [ SHARED ]
```
Use this command to retrieve the list of all Directory units created. If the SHARED keyword is used, the cluster-wide Units are listed.

This command produces an output - a dictionary, where the keys are Directory Unit mount points, and the values are Directory Unit names.

```
CREATEDIRECTORYUNIT unitName [ SHARED ] [ REMOTE ] mountPoint
```
Use this command to create a new Directory Unit.

> *unitName* : *string*
>> This parameter specifies the new Unit name.

> *mountPoint* : *string*
>> This parameter specifies the new Unit mount point (mount DN).

If the SHARED keyword is used, a cluster-wide Directory Unit is created.
If the REMOTE keyword is used, a Remote (LDAP-based) Directory Unit is created, otherwise a Local (File-based) Directory Unit is created.

```
RELOCATEDIRECTORYUNIT unitName [ SHARED ] newMountPoint
```
Use this command to re-mount an existing Directory Unit on a different mount point.

> *unitName* : *string*
>> This parameter specifies the Directory Unit name. If the SHARED keyword is used, this is a cluster-wide Directory Unit name.

> *mountPoint* : *string*
>> This parameter specifies the new mount point (mount DN).

```
DELETEDIRECTORYUNIT unitName [ SHARED ]
```
Use this command to remove an existing Directory Unit.

> *unitName* : *string*
>> This parameter specifies the Directory Unit name. If the SHARED keyword is used, this is a cluster-wide Directory Unit name.

```
GETDIRECTORYUNIT unitName [ SHARED ]
```
Use this command to retrieve the Directory Unit settings.

> *unitName* : *string*
>> This parameter specifies the Directory Unit name. If the SHARED keyword is used, this is a cluster-wide Directory Unit name.

This command produces an output - a dictionary with the Directory Unit settings.

```
SETDIRECTORYUNIT unitName [ SHARED ] newSettings
```
Use this command to change the Directory Unit settings.

> *unitName* : *string*
>> This parameter specifies the Directory Unit name. If the SHARED keyword is used, this is a cluster-wide Directory Unit name.

> *newSettings* : *dictionary*
>> This parameter specifies the new Directory Unit settings.

```
GETDIRECTORYACCESSRIGHTS [ SHARED ]
```
Use this command to retrieve the Directory Access Rights. If the SHARED keyword is used, the cluster-wide Access Rights

are retrieved.

This command produces an output - an array of Access Rights elements.

`SETDIRECTORYACCESSRIGHTS [ SHARED ] ` *`newAccessRights`*
Use this command to set the Directory Access Rights. If the SHARED keyword is used, the cluster-wide Access Rights are set.

*`newAccessRights`* : *`array`*
This parameter specifies the new Directory Access Rights.

# Miscellaneous Commands

`LISTCLICOMMANDS`
Use this command to retrieve the list of all CLI commands supported by this version of CommuniGate Pro Server.
This command produces an output - an *array* of strings, where each string is a supported command name.

`NOOP`
This command always completes successfully.

`ECHO` *`object`*
This command produces an output - an *object*, which is the command parameter copy.

`GETVERSION`
This command produces an output - a *string* with this CommuniGate Pro Server version.

`GETSYSTEMINFO` *`what`*
This command produces an output - an *object* returned by the CG/PL [SystemInfo](#) function called with the *what* parameter. If that function returns a null-object, this command returns an error.

`GETCURRENTTIME`
This command produces an output - a *timestamp* with this CommuniGate Pro Server internal timer value.

`SETLOGALL [ ON | OFF ]`
Use this command to switch on and off the "Log Everything" mode (this mode can also be enabled by using the `--LogAll` command line option.
To use this command, the user should have the "Can Monitor" Server Administration right.

`DUMPALLOBJECTS`
Use this command to write the list of all application data objects into the OS syslog.
**Note:** this list may contain millions of objects, and this command can easily overload the OS syslog facilities. It also blocks object creation and releasing functionality, effectively suspending CommuniGate Pro Server activities till all objects are listed.
To use this command, the user should have the "Can Monitor" Server Administration right.

`TESTLOOP` *`seconds`*
Use this command to test the server CPU load. The command executes some calculation loop for the specified number of seconds. This command produces an output - a *number* that indicates the average CLI thread CPU performance (the number of times the test loop was executed divided by the test time).
To use this command, the user should have the "Can Monitor" Server Administration right.

`SETTRACE` *`facility`* `[ ON | OFF ]`
Use this command to switch on and off internal logging facitilies that write to OS syslog. The *facility* parameter should be a string with one of the folloing supported values:

`FileIO`
record all file read/write/truncate operations

`FileOp`
record all file create/rename/remove operations

To use this command, the user should have the "Can Monitor" Server Administration right.

WRITELOG *logLevel logRecord*
    Use this command to store a record into the Server Log.

    *logLevel* : *number*
        This parameter specifies the record log level.

    *logRecord* : *string*
        This parameter specifies the string to be placed into the Server Log.

    Log records generated with this command have the SYSTEM prefix.
    To use this command, the user should have the "Can Monitor" Server Administration right.

RELEASESMTPQUEUE *queueName*
    Use this command to release an SMTP queue.

    *queueName* : *string*
        This parameter specifies the queue (domain) name to release.

    In a Dynamic Cluster environment this command releases the specified SMTP queue on all servers.
    To use this command, the user should have the "Can Monitor" Server Administration right.

REJECTQUEUEMESSAGE *messageID* [ REPORT *errorText* ]
    Use this command to reject a message from the Server Queue.

    *messageID* : *number*
        This parameter specifies the message ID.

    *errorText* : *string*
        This optional parameter specifies the text to be included into the error report (bounce) sent to the message sender. If this parameter is NONDN, no DSN report message is generated.

    To use this command, the user should have the "Can Reject Queues" Server Administration right.

REJECTQUEUEMESSAGES SENDER *authedSender* [ REPORT *errorText* ]
    Use this command to reject all messages sent by the specified sender from the Server Queue.

    *authedSender* : *string*
        This parameter specifies the authenticated sender's name.

    *errorText* : *string*
        This optional parameter specifies the text to be included into the error report (bounce) sent to the message sender. If this parameter is NONDN, no DSN report message is generated.

    In a Dynamic Cluster environment this command rejects messages from all server queues.
    To use this command, the user should have the "Can Reject Queues" Server Administration right.

GETMESSAGEQUEUEINFO *moduleName* QUEUE *queueName*
    Use this command to read information about a module message Queue.

    *moduleName* : *string*
        This parameter specifies the module name.

    *queueName* : *string*
        This parameter specifies the module queue name.

    This command produces an output - a *dictionary* with the specified queue information.

If the module does not have the specified queue, the dictionary is empty. Otherwise it contains the following elements:

nTotal
>    a number - the total number of messages in the queue

size
>    a number - the total size of all messages in the queue

delayedTill
>    (optional) a timestamp - the effective release time for this queue

lastError
>    (optional) a string with the last problem report

GETCURRENTCONTROLLER
>    Use this command to get the IP address of the current Dynamic Cluster Controller.
>    This command produces an output - a *string* with the Cluster Controller IP Address.
>    To use this command, the user should have the "Can Monitor" Server Administration right.

RECONNECTCLUSTERADMIN
>    Use this command to force a Dynamic Cluster member to re-open all its inter-cluster Administrative connections, and (for a non-controller member) to re-open its Administrative connection to the Controller.

GETTEMPCLIENTIPS
>    Use this command to retrieve the set of temporary Client IP Addresses. The command produces an output - a *string* with Temporary Client IP addresses separated with the comma (,) symbols.
>    To use this command, the user should have the "Can Monitor" Server Administration right.

REPORTFAILEDLOGINADDRESS *address*
>    Use this command to increment the counter of failed Login attempts from the specified IP address used in the [Temporarily Blocked Addresses](#) functionality.

>    *address* : *string*
>>        The Network IP Address to report.

>    To use this command, the user should have the "Server Settings" Server Administration right.

TEMPBLACKLISTIP *address* [ TIMEOUT *seconds* | DELETE ]
>    Use this command to add an address to the Temporary Blacklisted IP Addresses set.

>    *address* : *string*
>>        The Network IP Address to add.

>    *seconds* : *number*
>>        The time period the address should be blacklisted for.

>    Use the DELETE keyword or specify zero time period to remove the address from the Temporary Blacklisted IP Addresses set.

GETTEMPBLACKLISTEDIPS
>    Use this command to retrieve the set of Temporary Blacklisted IP Addresses. The command produces an output - a *string* with Temporary Blacklisted IP addresses separated with the comma (,) symbols.
>    Each IP address may have a -*nnnn* suffix, where nnnn is either the number of seconds this address will remain blacklisted for, or the * symbol indicating permanent address blacklisting.
>    To use this command, the user should have the "Can Monitor" Server Administration right.

SETTEMPBLACKLISTEDIPS *addresses*
>    Use this command to add addresses to the Temporary Blacklisted IP Addresses list.

>    *addresses* : *string*
>>        A string with a list of IP addresses, using the output format of the `GetTempBlacklistedIPs` command.

To use this command, the user should have the "Server Settings" Server Administration right.

---

# Index

GETMAILBOXSUBSCRIPTION, SETMAILBOXSUBSCRIPTION

GETMAILBOXALIASES, SETMAILBOXALIASES

## Alert Administration

GETDOMAINALERTS, SETDOMAINALERTS, POSTDOMAINALERT, REMOVEDOMAINALERT

GETACCOUNTALERTS, SETACCOUNTALERTS, POSTACCOUNTALERT, REMOVEACCOUNTALERT

GETSERVERALERTS, SETSERVERALERTS, POSTSERVERALERT, REMOVESERVERALERT, GETCLUSTERALERTS, SETCLUSTERALERTS, POSTCLUSTERALERT, REMOVECLUSTERALERT

## File Storage Administration

READSTORAGEFILE, WRITESTORAGEFILE, RENAMESTORAGEFILE, DELETESTORAGEFILE, LISTSTORAGEFILES, GETSTORAGEFILEINFO READSTORAGEFILEATTR UPDATESTORAGEFILEATTR GETFILESUBSCRIPTION SETFILESUBSCRIPTION

## Mailing Lists Administration

LISTLISTS, GETDOMAINLISTS, GETACCOUNTLISTS

CREATELIST, RENAMELIST, DELETELIST, GETLIST, UPDATELIST

LIST, LISTSUBSCRIBERS, READSUBSCRIBERS

GETSUBSCRIBERINFO, SETPOSTINGMODE, PROCESSBOUNCE

## Web Skins Administration

LISTDOMAINSKINS, CREATEDOMAINSKIN, RENAMEDOMAINSKIN, DELETEDOMAINSKIN, LISTDOMAINSKINFILES, READDOMAINSKINFILE, STOREDOMAINSKINFILE

LISTSERVERSKINS, CREATESERVERSKIN, RENAMESERVERSKIN, DELETESERVERSKIN, LISTSERVERSKINFILES, READSERVERSKINFILE, STORESERVERSKINFILE

LISTCLUSTERSKINS, CREATECLUSTERSKIN, RENAMECLUSTERSKIN, DELETECLUSTERSKIN, LISTCLUSTERSKINFILES, READCLUSTERSKINFILE, STORECLUSTERSKINFILE, STORECLUSTERSKINFILE

LISTSTOCKSKINFILES, READSTOCKSKINFILE

## Web Interface Integration

CREATEWEBUSERSESSION, CREATEXIMSSSESSION, FINDACCOUNTSESSION, LISTACCOUNTSESSIONS, GETSESSION, KILLSESSION, BLESSSESSION

## Real-Time Application Administration

CREATEDOMAINPBX, DELETEDOMAINPBX, LISTDOMAINPBXFILES, READDOMAINPBXFILE, STOREDOMAINPBXFILE

CREATESERVERPBX, DELETESERVERPBX, LISTSERVERPBXFILES, READSERVERPBXFILE, STORESERVERPBXFILE

CREATECLUSTERPBX, DELETECLUSTERPBX, LISTCLUSTERPBXFILES, READCLUSTERPBXFILE, STORECLUSTERPBXFILE

LISTSTOCKPBXFILES, READSTOCKPBXFILE

## Real-Time Application Control

STARTPBXTASK, SENDTASKEVENT, KILLNODE, READNODESTATUS

## Account Services

REMOVEACCOUNTSUBSET DATASET ROSTER BALANCE

## Server Settings

LISTMODULES, GETMODULE, SETMODULE, UPDATEMODULE

GETQUEUESETTINGS, SETQUEUESETTINGS, GETSIGNALSETTINGS, SETSIGNALSETTINGS, GETMEDIASERVERSETTINGS, SETMEDIASERVERSETTINGS

GETSESSIONSETTINGS, SETSESSIONSETTINGS

GETCLUSTERSETTINGS, SETCLUSTERSETTINGS

GETLOGSETTINGS, UPDATELOGSETTINGS

GETNETWORK, SETNETWORK, GETCLUSTERNETWORK, SETCLUSTERNETWORK

GETDNRSETTINGS, SETDNRSETTINGS,

GETBANNED, GETCLUSTERBANNED, SETBANNED, SETCLUSTERBANNED

GETSERVERMAILRULES, SETSERVERMAILRULES, GETCLUSTERMAILRULES, SETCLUSTERMAILRULES

GETSERVERSIGNALRULES, SETSERVERSIGNALRULES, GETCLUSTERSIGNALRULES, SETCLUSTERSIGNALRULES

GETROUTERTABLE, SETROUTERTABLE, GETCLUSTERROUTERTABLE, SETCLUSTERROUTERTABLE

GETROUTERSETTINGS, SETROUTERSETTINGS, GETCLUSTERROUTERSETTINGS, SETCLUSTERROUTERSETTINGS

REFRESHOSDATA

GETLANIPS, SETLANIPS, GETCLUSTERLANIPS, SETCLUSTERLANIPS

GETCLIENTIPS, SETCLIENTIPS, GETCLUSTERCLIENTIPS, SETCLUSTERCLIENTIPS

GETBLACKLISTEDIPS, SETBLACKLISTEDIPS, GETCLUSTERBLACKLISTEDIPS, SETCLUSTERBLACKLISTEDIPS

GETWHITEHOLEIPS, SETWHITEHOLEIPS, GETCLUSTERWHITEHOLEIPS, SETCLUSTERWHITEHOLEIPS

GETNATEDIPS, SETNATEDIPS, GETCLUSTERNATEDIPS, SETCLUSTERNATEDIPS

GETNATSITEIPS, SETNATSITEIPS, GETCLUSTERNATSITEIPS, SETCLUSTERNATSITEIPS

GETDEBUGIPS, SETDEBUGIPS, GETCLUSTERDEBUGIPS, SETCLUSTERDEBUGIPS

GETDENIEDIPS, SETDENIEDIPS, GETCLUSTERDENIEDIPS, SETCLUSTERDENIEDIPS

ROUTE GETIPSTATE

GETSERVERINTERCEPT, SETSERVERINTERCEPT, GETCLUSTERINTERCEPT, SETCLUSTERINTERCEPT
GETSERVERSETTINGS UPDATESERVERSETTINGS

Monitoring
GETSTATELEMENT, SETSTATELEMENT, GETNEXTSTATNAME, GETDIALOGINFO, SHUTDOWN

Statistics
GETACCOUNTSTAT, RESETACCOUNTSTAT, GETDOMAINSTAT, RESETDOMAINSTAT

Directory Administration
LISTDIRECTORYUNITS, CREATEDIRECTORYUNIT, RELOCATEDIRECTORYUNIT, DELETEDIRECTORYUNIT, GETDIRECTORYUNIT, SETDIRECTORYUNIT, GETDIRECTORYACCESSRIGHTS, SETDIRECTORYACCESSRIGHTS

Miscellaneous Commands
LISTCLICOMMANDS, NOOP, ECHO, GETVERSION, GETSYSTEMINFO, GETCURRENTTIME, SETLOGALL, DUMPALLOBJECTS, TESTLOOP, WRITELOG, RELEASESMTPQUEUE, REJECTQUEUEMESSAGE, REJECTQUEUEMESSAGES, GETMESSAGEQUEUEINFO, GETCURRENTCONTROLLER, RECONNECTCLUSTERADMIN, GETTEMPCLIENTIPS, REPORTFAILEDLOGINADDRESS, TEMPBLACKLISTIP, GETTEMPBLACKLISTEDIPS, SETTEMPBLACKLISTEDIPS,

---

Applications **Data** **CLI/API** **Rules** **CG/PL** **PBXApp** **XIMSS** **WebApp** **WSSP** **Helpers**

# Automated Rules

- **Specifying Rules**
- **Creating, Renaming and Removing Rules**
- **Rule Conditions**
    - **String Lists**
- **Rule Actions**
- **Domain-wide Rules**
- **Cluster-wide Rules**

The CommuniGate Pro Server can automatically process E-mail Messages and Signals using sets of Automated Rules.

The Server-wide and Cluster-wide sets of Automated Rules are applied to all Messages and to all Signals transferred via the Server or the Cluster.

The Account-level sets of Automated Rules are applied to all E-mail Messages and to all Signals sent to the Account.

Each Rule in a set has a name, a priority, a set of conditions, and a set of actions. The higher priority Rules are checked first: a Rule with the priority level of 9 is applied before a Rule with the priority level 1.

If a Message or a Signal meets all Rule conditions, the Rule actions are performed, and automated processing either stops, or proceeds checking other, lower-priority Rules.

# Specifying Rules

System administrators can specify Server-Wide and Cluster-Wide Rules.

To specify Server-Wide Message (Queue) Rules, use the WebAdmin Interface to open the Mail pages in the Settings realm, and click the Rules link.
To specify Server-Wide Signal Rules, use the WebAdmin Interface to open the Real-Time pages in the Settings realm, and click the Rules link.

System and Domain Administrators can specify Account Rules using links on the Account Settings page.

Account users can specify their Rules themselves, using the WebUser Interface. System or Domain administrators can limit the set of Rule actions a user is allowed to specify.

System and Domain Administrators can specify Domain-Wide Rules using the Rules links on the Domain Settings page.

---

# Creating, Renaming and Removing Rules

When the list of Rules appears in a browser window, the Rule names and priorities can be modified:

| Priority | Name |
| --- | --- |

| | |
|---|---|
| 7 | Edit |
| 2 | Edit |
| Inactive | Edit |

After you have modified the Rule names and/or priorities, click the Update button. The list is displayed re-sorted by priority.

Rules with the `Inactive` priority are not applied, but they are not deleted from the Rule set, and they can be re-enabled at any moment.

To create a new Rule, enter its name in the field on the top and click the Add Rule button.

To remove a Rule, select the checkbox in the Delete column and click the Update button.

To modify the Rule conditions and actions, click the Edit link.

---

# Rule Conditions

Each Rule can have zero, one, or several conditions. The conditions are checked in the same order they are specified. If an E-mail Message or a Signal meets all the Rule conditions, the Rule actions are performed.

The condition operations `is` and `is not` process their parameters as "pictures": the asterisk (`*`) symbols in parameters are processed as wildcards that match zero or more symbols in the tested string. To check that a string contains the `@thatdomain` substring, the `is *@thatdomain*` operation should be used, and to check that a string does not end with the `somedomain.com` substring, the `is not *somedomain.com` operation should be used.

The condition operations `in` and `not in` process their parameters as sets of one or more "pictures" separated with the comma (`,`) symbols. The tested string is compared to all picture strings. The `in` condition is met if the tested string matches at least one picture string. The `not in` condition is met if the tested string does not match any picture string in the specified set.
**Note:** do not use excessive spaces around the comma signs: spaces before the comma sign become trailing spaces of the previous picture, and spaces after the comma sign become leading spaces of the next picture.

The following Rule conditions can be used in both E-Mail Queue and Signal processing Rules:

`Submit Address [is | is not | in | not in]` *string*
>   This condition checks the E-mail or Signal submit address.
>   If the Message or Signal was generated within the Server itself, its submit address is empty.
>   Otherwise the "submit address" string contains the name of the component that received or generated the E-mail Message or Signal, and (separated with a space symbol) the network (IP) address the E-mail or Signal came from, optionally followed with the IP port number (separated with a colon symbol).
>   If the Message or Signal was generated locally, with an internal Server component (such as Rules), the network address field contains `[0.0.0.0]`.

>   Sample (Queue Rule):

| | |
|---|---|
| Submit Address | is |

>   Sample (Signal Rule):

| | |
|---|---|
| Submit Address | is |

`Time Of Day [is | is not | less than | greater than | in | not in]` *time string*
>   This condition checks the current time of day in the user's time zone (for the Account-level Rules) or in the Server Time Zone (for the system-wide Rules).

This condition allows you to compose rules that are applied only at certain times of day.

A time string should be specified as `hh:mm` or `hh:mm:ss`, where *hh* is the hour, *mm* - minutes, *ss* - seconds. Time strings can contain the `am` or `pm` suffix.

If the condition is `in` or `not in`, then the parameter string should contain a pair of time strings, separated with the minus (`-`) symbol.
If the second time value is not smaller than the first one (as in `08:30-5:15pm`), the `in` condition is met at any time after 8:30 and before 17:15.
If the second time value is smaller than the first one (as in `22:30-5:15`), the `in` condition is met at any time after 22:30 and at any time before 5:15.

Sample (Queue Rule):

| | |
|---|---|
| Time Of Day | greater than |
| Time Of Day | in |

If the Account-level Rule condition is `is`, `is not`, `in`, or `not in` the `work` string can be used as *time string*.
If the current day of week is included into the "working days" set and the current time of day is inside the "working hours" range specified for this Account, then the `is` and `in` conditions are met. Otherwise, the `is not` and `not in` conditions are met.

`Current Date [is | is not | less than | greater than] date string`
This condition checks the current time and date. This condition allows you to compose rules that are applied to Messages or Signals only before or after the specified date and time.

To compare dates only, specify the date string in the following format:

- `DD MMM YYYY`

To compare date and time, specify the date string using one of the following formats:

- `DD MMM YYYY hh:mm`
- `DD MMM YYYY hh:mm:ss`
- `DD MMM YYYY hh:mm:ss +ZZZZ`
- `DD MMM YYYY hh:mm:ss -ZZZZ`

where:

`DD` is the day of month

`MMM` is month specified as a 3-letter English abbreviation:
`Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec`

`YYYY` is the year

`hh` is the hour

`mm` is the minute

`ss` is the second

`+ZZZZ` or `-ZZZZ` is the time zone; if the time zone is not specified, the user-specified time zone is used for the Account-level Rules and the Server time zone is used for the Server-wide and Cluster-wide Rules.

Sample:

| Current Date | greater than |
|---|---|
| Current Date | less than |

**Current Day** [is | is not | in | not in] *day string*

This condition checks the current day of week. It allows you to compose rules that are applied to Messages or Signals only on certain days of week.

Days should be specified either as numbers (`0` for Sunday, `6` for Saturday), or as RFC822 abbreviations (`Mon`, `Tue`, `Wed`, `Thu`, `Fri`, `Sat`, `Sun`).

Sample:

| Current Day | in |
|---|---|

**Preference** [is | is not | in | not in] *string*

The *string* must contain the Account Preference name and the preference data picture, separated with the colon (`:`) symbol. This condition compares the specified Account Preference value with the data picture.

Sample:

| Preference | is |
|---|---|

**FreeBusy** [is | is not | in | not in] *status string*

This condition should not be used in the Server-wide or Cluster-wide Rule. The condition retrieves the Account Free-Busy data and retrieves the status value for the current time. This status value is compared to the specified *status string*.

Sample:

| FreeBusy | is |
|---|---|

**Existing Mailbox** [is | is not] *string*

The parameter specifies a mailbox name, and this condition checks if the specified Mailbox exists (or if it does not exist). A Mailbox "exists" if it is possible to open the Mailbox with the specified name and to add a Message to it. If this condition is used in an Account-level Rule and the parameter specifies a Mailbox in a different Account, and that Mailbox exists, but the current user cannot add a message to it, the Mailbox is treated as one that "does not exist" for this Rule condition.

Sample:

| Existing Mailbox | is |
|---|---|

This condition is useful in Domain-Wide Rules: a Rule can check if the current Account has a special Mailbox, and copy certain Messages to that Mailbox only if it exists.

## String Lists

The CommuniGate Pro Server can store named lists of strings as the [Account DataSet](#) subsets.

Each list can contain zero, one, or several strings. The Rule Condition operations can refer to those lists, if:

- The Rule is an Account-level (or a Domain-wide) one.
- The condition operation is `in` or `not in`.
- The operation parameter is specified as a *string*.
- The operation parameter starts with the hash (`#`) symbol.

For example, the Condition operation

```
Sender    in     #Blocked
```

checks if the E-mail Message or Signal sender's address is included into the String List called `Blocked`.

String List subsets can be used as WebUser Interface [Address Books](#).

---

# Rule Actions

Each Rule can have zero, one, or several actions. If an E-mail Message or Signal meets all the Rule conditions, the Rule actions are performed.

Rule Action parameters can contain Macro symbol combinations (`^x`, where *X* is a letter).
Different sets of Macro symbol combinations are processed
See the [Signal Rules](#) and [E-mail Rules](#) sections to learn about the Macro symbol combinations available.

The following Rule actions can be used in all Message and Signal processing Rules:

`Reject` *errorMessage*
    This action should be the last one in a Rule. Execution of this Rule stops and no other (lower-priority) Rules are checked.
    If a Rule is a Signal processing one, the Signal fails with the specified error code. If there are Signal Rules for the "Failure" condition, those Rules are applied, otherwise the the Signal is rejected immediately.
    If a Rule is an E-mail processing one, the Message is rejected, and a negative Delivery Notification is sent back to the Message sender.
    If the action parameter text is not empty, it is used as the error report text.
    E-mail processing Rules can still store rejected Messages using the Store action before the Reject action.

    Sample:

| Subject | is |
|---|---|

| Reject with | |
|---|---|

`SendURL` *URL* (Execute URL *URL*)
    An HTTP connection is made to the remote Web server specified in the URL, and an HTTP GET request with the specified URL is sent to that server.
    If any response is received, the response is discarded. If no response is received within 10 seconds, the HTTP connection is closed.
    When the *URL* parameter is processed using [Macro Substitution](#), the calculated values are URL-encoded first.

    Sample:

| Execute URL | |
|---|---|

```
Send IM messageText
```
An Instant message with the specified message text is sent.
If the message text starts with the

```
        To: user@domain
```

line, then the Instant Message is sent to the specified address. Otherwise, the message is sent to the current Account (for Account-level Rules), or it is not sent at all (for Server-wide/Cluster-wide Rules).

Sample:

> Send IM

```
FingerNotify [ address ]
```
The Server connects to the computer at the specified network address, port 79 (the *finger* port), and sends the `nm_notifyuser` string to that computer. If the address is not specified, and the action is executed as a part of an Account-Level Rule, the network address of the last user Login is used.
This action can be used with the Finger-based utilities (such as [NotifyMail®](#)) installed on client computers.

Sample:

> FingerNotify

Users are allowed to specify this action only if they are allowed to specify `execute`-type actions.

System Administrators can use the WebAdmin Interface to configure the Notifier settings. Open the General pages in the Settings realm, and find the Notifier panel on the Others page:

| Notifier | |
|---|---|
| Log Level:   Failures | Queue Size:   100 |

```
Write To Log recordText
```
A Major-Level (Level 2) record with the Message or Signal ID and the specified *recordText* string is placed into the [System Log](#).
Only the Server Administrator is allowed to specify this action.

```
Remember 'From' in stringList
```
This action can be used in Account-Level Rules only. The operation parameter specifies the name of a string list that exists in or should be created in the Account dataset. The Message author or Signal sender (`From`) address is added to the specified list.
If the list already has 500 or more elements, the new element is not added.

# Domain-Wide Rules

[Domain Administrators](#) can specify Domain-wide Rules.

When a Message is being delivered to any Account by the Local Delivery module, or when a Signal targets any Server Account, the "effective" set of Account-level Rules is applied.

The first Rules in the effective set are Domain-wide Rules with priorities above 5, then it includes all Account-level Rules, and then - all Domain-wide Rules with priorities equal to or less than 5.

This method guarantees that all Domain-Wide Rules with priorities higher than 5 are applied before any Account Rule. If such a Domain-Wide Rule uses the Stop Processing action, no Account Rules are applied.

**Note:** Domain-Wide Rules are "mixed" with the Account Rules and are applied in the same environment as the Account Rules, "on behalf" of the Account user.

# Cluster-Wide Rules

The Dynamic Cluster Administrators can see an additional link on the Rules pages of the WebAdmin Interface. This link can be used to open the list of Cluster-wide Rules.

When you modify the Cluster-wide Rules set on any Cluster Member, the set is automatically updated on all Cluster members.

The effective set of "server-wide" rules for each Cluster member is a union of the Server-Wide Rules explicitly set on that Cluster member and the Cluster-wide Rules.

Rules from both sets are applied together, in the order specified with the Rule priority attribute. For example, Messages can be processed with a high-priority Cluster-wide Rule, then with a medium-priority Server-wide Rule, then with a low-priority Cluster-wide Rule.

# CommuniGate Programming Language (CG/PL)

- **Data Model**
- **Lexemes**
- **Literals**
- **Code Sections**
  - **Modules**
  - **Entries**
  - **Procedures**
  - **Functions**
- **Variables**
- **Constants**
- **Expressions**
- **Operators**
- **Built-in Procedures and Functions**
  - **Strings**
  - **Numbers**
  - **TimeStamps**
  - **IP Addresses**
  - **Datablocks**
  - **Arrays**
  - **Dictionaries**
  - **XML Objects**
  - **Data Conversion**
  - **Cryptography**
  - **Environment**
  - **Addresses and URIs**
  - **Account Data**
  - **Mailboxes**
  - **Mailbox Handles**
  - **Message Handles**
  - **Calendars**
  - **File Storage**
  - **Roster**
  - **Datasets**
  - **Directory**
  - **Services**
  - **Communications**
- **Multitasking**
  - **Spawning**
  - **Events**
  - **Meetings**
  - **Queues**
- **Formal Syntax**

The CommuniGate Programming Language (CG/PL) is a powerful, yet simple procedural language. It can be used with various components of the CommuniGate Pro Server software, including:

- Real-Time Applications (such as "IP-PBX").
- Web Applications.

The language structure and the language features are the same for all applications, but different applications use different sets of built-in functions and procedures.

The following is a simple CG/PL program (using the "basic" and "alternative" syntax):

```
//
// A simple CG/PL application
//
entry Main is
  myName = "Jim" + " " + "Smith";
  if length(myName) > 10 then
    myName = Substring(myName,0,8) + "..";
  end if;
end;
```

```
/*
 * A simple CG/PL application
 */
entry Main {
  myName = "Jim" + " " + "Smith";
  if(myName.length() > 10) {
    myName = myName.substring(0,8) + "..";
  }
}
```

# Data Model

Information is presented as *objects*. An object can be a string, a number, a datablock, a timestamp, an ip-address, an array, a dictionary, or an XML object. See the Data section for more details.

An object can also be a *Mailbox handle*, a *Task handle*, or any other special object defined within a particular environment.

A special type of object is the *null-value* (null value or "no object" value).

When processing logical (*boolean*) values, the *null-value* is used as a false-value, and a string "YES" is used as a *true-value*.

# Lexemes

The program *source code* is a plain text encoded using the UTF-8 character set.

The language lexemes are:

- *keywords* - **and**, **by**, **elif**, **else**, **entry**, **exitif**, **external**, **false**, **for**, **forward**, **function**, **if**, **is**, **not**, **loop**, **null**, **or**, **procedure**, **return**, **spawn**, **stop**, **then**, **true**, **var**, **xor**, **while**.
  All keywords are case-insensitive.
- *names* - sequences starting with alpha-symbols and optionally followed by alpha-symbols, digits, and/or underscore symbols.
  All names are case-insensitive.
  The keywords cannot be used as names.
- *signs* - **+, -, *, //, /, %, +=, -=, *=, /=, %=, =, ==, !=, <, <=, >, >=, !, &, &&, &=, |, ||, |=, ?, ;, ,, :, (, ), [, ], {, }**
- *numbers* - sequences of digits.
- *strings* - quoted strings as specified in the Data section.

A comment is a double-slash (`//`) symbol and all following symbols up to and including the next EOL (end-of-line) symbol.

A comment is a pair of the slash and star (`/*`) symbols and all following symbols up to and including the first pair of the star and slash (`*/`) symbols.

*white-space* is a sequence of 1 or more space symbols, tabulation symbols, EOL symbols, and/or comments. At least one white space is required between a name and a keyword, other lexemes can be separated with zero or more white spaces.

# Literals

Literals are constant values representing themselves. A literal is one of the following:

- a *number* lexem
- a *string* lexem
- the *null* keyword (the null-value)
- the *false* keyword (the false-value, same as the null-value)
- the *true* keyword (the true-value)

---

# Code Sections

A program code is a set of code sections:

- Entries are code sections executed when the program is activated.
- Procedures and functions are code sections that can be used in (*called from*) entry code sections, and other procedures and/or functions.

The procedures and functions can be called *recursively*: a procedure or a function can call itself - directly or via calling some other procedures or functions.

All code sections are *reenterable*: the same code section can be used by several program *activations* or *Tasks* at the same time.

Code sections must be declared before they can be used. Declarations include forward-declarations, external-declarations, and definitions.

A program code may contain not more than one forward-declaration of a code section, specified before its definition. Forward-definitions are used in programs containing two or more code sections calling each other, so it is not possible to define each section before it is used in the other code section.

## Modules

External-declarations allow code sections to call code sections defined in separate program code *modules*.
External-declarations are allowed only in the environments that support them, such as the [Real-Time Application](#) and [Web Applications](#) environments. See the environment description for more details.
If the external-declaration specifies the code section name as a regular name, when this code section is used, the module with the same name as the code section name is loaded: if the external declaration is
```
function myName(param) external;
```
and the myName function is called, then the separate program code module "myName" is loaded, and its code section "myName" is called.
The external-declaration may contain a qualified name, explicitly specifying the module name: if the external declaration is
```
function myModule::myName(param) external;
```
and the myModule::myName function is called, then the separate program code module "myModule" is loaded, and its code section "myName" is called.
The code section name and its parameter names specified in an external-declaration must match the code section name and its parameter names specified in the code section definition given in an external program code *module*.
The code section definition can specify more parameters than an external-declaration. The missing parameters are assigned null-values when such an external-declaration is used.

A program code should contain exactly one external-declaration or exactly one definition (but not both) of each code section used.

## Entries

An entry declaration consists of the `entry` keyword, the entry name, followed by one of the following:

for a forward-declaration:
    the `forward` keyword followed by the semicolon (`;`) symbol
for an entry definition:
    the `is` keyword followed by an operator sequence, followed by the `end` keyword (optionally followed by the `entry` keyword), and followed by the semicolon (`;`) symbol

for an alternative entry definition:
  the left brace (`{`) symbol followed by an operator sequence, followed by the right brace (`}`) symbol

If a running program reaches the end of an entry section operator sequence, an implicit `stop` operator is executed.

The following example shows an entry for a [Real-Time Application](). It tries to accept an incoming call, and if succeeds, it plays a sound. The entry code ends, quitting the program and thus finishing the call:

```
entry main is
  if acceptCall() == null then
    playFile("greeting.wav");
  end if;
end entry;
```

Or, in the alternative form, and using the alternative form of the conditional operator:

```
entry main {
  if (acceptCall() == null) {
    playFile("greeting.wav");
  }
}
```

## Procedures

A procedure declaration consists of the **procedure** keyword, the procedure name, the left parenthesis, an optional list of parameter names, the right parenthesis, followed by one of the following:

for a forward-declaration:
  the **forward** keyword followed by the semicolon (`;`) symbol
for a procedure definition:
  the **is** keyword followed by an operator sequence, followed by the **end** keyword (optionally followed by the **procedure** keyword), and followed by the semicolon (`;`) symbol
for an alternative procedure definition:
  the left brace (`{`) symbol followed by an operator sequence, followed by the right brace (`}`) symbol
for an external-declaration:
  the **external** keyword followed by the semicolon (`;`) symbol

If a running program reaches the end of a procedure operator sequence, an implicit `return` operator is executed.

The following example shows a [Real-Time Application]() procedure that speaks the specified number:

```
procedure sayNumber(x) is
  x = x % 100;
  if x >= 10 then
    PlayFile(String(x/10*10)+".wav");
  end if;
  if x != 0 then
    PlayFile(String(x%10)+".wav");
  end if;
end procedure;
```

If a forward-declaration is used, the procedure definition must have exactly the same parameters as its forward-declaration.

Example:

```
procedure sayMoney(x,units) forward;
procedure sayNumber(x) forward;
procedure sayMoney(x,units) is
  sayNumber(x);
  PlayFile(units+".wav");
end procedure;
```

## Functions

A function declaration is the same as a procedure declaration, but the **function** keyword is used instead of the **procedure** keyword.

All program control paths in a function code section must end with a `return` or a `stop` operator.

The following example defines a function calculating the factorial of its argument:

```
function Factorial(x) is
   if x <= 1 then return 1; end if;
   return Factorial(x-1)*x;
end function;
```

Below is the same function definition in the alternative form, using the ternary operator:

```
function Factorial(x) {
   return x <= 1 ? 1 : Factorial(x-1)*x;
}
```

All code section (entry, procedure, and function) names used within the same program code must be unique.

---

# Variables

Variables are containers. Any object (including a null-value) can be assigned to a variable, and that object becomes the variable value. Initially, the variable value is a null-value.

Local variables are local to the code section (function/procedure/entry) instance (invocation) where they were created. All their values are destroyed when the code section exits (returns).

To declare variables, use the `var` keyword and the list of variable names:

```
var var1,var2,var3;
```

It is possible to declare a local variable and immediately assign it some value, using an initial value expression:

```
var var1=expr1,var2,var3=expr3;
```

Each code section can contain several `var` declarations. Variables should be used after they are declared (later in the code section code).

```
var myCount;            // the myCount variable is created
myCount = 3;            // the numeric value 3 is assigned to it
myCount = myCount + 2;  // myCount variable value changed to 5
myCount = "Service";    // myCount variable value changed to a string
```

Alternatively, if the code section contains no explicit `var` declaration, any name used in any code section operator which is not declared as a [procedure or function](#) name, and which is not matching a [built-in](#) procedure or function name declares a variable with this name.

```
myCount = 3;            // the myCount variable is created and
                        // the numeric value 3 is assigned to it
myCount = myCount + 2;  // myCount variable value changed to 5
myCount = "Service";    // myCount variable value changed to a string
```

**Note:** declaring a variable inside some "block operator" (such as the `if` or `loop` operator) declares this variable only inside this "block operator".

"Task variables" are stored in a dictionary unique for each Task ("invocation") created, and they can be used in all code sections, after being declared as Task variables. The Task variables dictionary is destroyed when the Task completes.

The Task variables are declared using the same `var` keyword and the variable name list, but these declarations should be placed outside any code section, and they should not contain any initial value expressions.

The alternative way to use the "task variables" is to access the Task variable dictionary directly, using the [Vars()](#) built-in function.

---

# Constants

Constants are names associated with literal values.

To declare constants, use the `const` keyword and the list of constant names and their values:

```
const c1=1234,c2="test",cFlag=true;
```

When the constant name is used, the value associated with it is substituted.

---

# Expressions

The language implements unary operations. Unary operations are specified as

*operation operand*

The following unary operations are supported:

**-**

unary minus. If the operand value is a number, operation value is the number with the opposite sign, otherwise the operation value is the number 0.
The following expressions have the value of `-5`:

```
-5
-(3+2)
```

**+**

unary plus. If the operand value is a number, operation value is that number, otherwise the operation value is the number 0.
The following expressions have the value of `5`:

```
+5
+(10/2)
```

**not**

Negation. If the operand value is a null-value, the operation value is a true-value, otherwise the operation value is a null-value.
The `!` symbol can be used instead of the **not** keyword.
The following expressions have the value of `"YES"`:

```
not null
!(2 == 3)
```

The unary operations have a higher priority than binary and ternary operations.

The following expression has the value of `-1`, not `-3`:

```
-2 + 1
```

The language implements binary operations. Binary operations are specified as

*left-operand operation right-operand*

Unless explicitly specified otherwise, both operands are computed first (in a non-specified order), and the operation is applied to the operand values.

The following binary operations are supported, listed in the descending priority order:

**\*, /, %**

These arithmetic operations can be applied to numeric operand values only.
If one of the operands is not a Number, or if the right operand of the `/` or `%` operations is number 0, the operation value is a null-value.
The following expressions have the value of `10`:

```
5 * 2
-20 / -2
30 % 20
```

**+, -**

These arithmetic operations can be applied to numeric operand values.

The `+` operation can be applied to a pair of string operand values, producing a string value with their catenation.

The `+` operation can be applied to a pair of datablock operand values, producing a datablock value with their catenation.

The `+` operation can be applied to a datablock and a string value operands, producing a datablock copy with the string bytes added to the datablock end.

The `+` operation can be applied to a datablock and a numeric value operands, where the numeric value should be in the 0..255 range, producing a datablock copy with an additional byte (containing the numeric operand value) added to the datablock end.

The `+` operation can be applied to a timestamp and a numeric value operands, producing a timestamp value that is the numeric value seconds later than the timestamp operand value.

The `-` operation can be applied to a left operand with a timestamp value and a right operand with a numeric value, producing a timestamp value that is the numeric value seconds earlier than the timestamp operand value.

The `-` operation can be applied to a pair or operands with timestamp values, producing a numeric value - the number of seconds between the left and the right operand values.

In other cases the operation value is a null-value.

The following expressions have the value of `5`:

```
3 + 2
-2 - -7
```

The following expression has the value of `"John Doe"`:

```
"Joh" + "n Doe"
```

**<, <=, ==, !=, >=, >**

These comparison operations can be applied to a pair of numeric operands, to produce a true-value when the arithmetic comparison condition is met.

These comparison operations can be applied to a pair of timestamp operands, to produce a true-value when the arithmetic comparison condition is met.

These comparison operations can be applied to a pair of string operands, to produce a true-value when the arithmetic comparison condition is met. Strings are interpreted as sequences of UTF-8 encoded 16-bit Unicode symbols, compared symbol by symbol.

These comparison operations can be applied to a pair of datablock operands, to produce a true-value when the arithmetic comparison condition is met. Datablocks are interpreted as sequences on bytes (with values in the 0..255 range), compared byte by byte.

The `==`, `!=` comparison operations can also be applied to any pair of objects, checking if they are "equal":

- A null-value is equal to and only to a null-value.
- Numbers are equal if their numeric values are equal.
- Strings are equal if they have the same length and have the same symbol in each string position.
- Datablocks are equal if they have the same length and have the same data byte in each datablock position.
- Arrays are equal if they have the same number of elements and elements in each position are equal.
- Dictionaries are equal if they have the same number of key-value pairs, the key strings are equal, and the values for each key are equal in both dictionaries.
- For all other object types any object is equal to itself only (no two different objects are considered equal).

In other cases the comparison operation value is a null-value.

The following expressions have the value of `"YES"`:

```
1+2 == 3
2+2 != 3
2+2 >= 3
"Joe" == "Joe"
null == (2 == 3)
```

**and, or, xor, and then, or else**

these logical operations compare their operands with null-values.

The **and** operation value is a true-value if both operand values are not null-values, and a null-value otherwise.

The **&** symbol can be used instead of the **and** keyword.

The **or** operation value is a true-value if at least of the operand values is not a null-value, and a null-value otherwise.

The **|** symbol can be used instead of the **or** keyword.

The **xor** operation value is a null-value if none or both operand values are null-values. Otherwise, the operation value is the operand value that is not a null-value.

The **^** symbol can be used instead of the **xor** keyword.

The **and then** operation computes the left operand value first. If that value is a null-value, the right operand is not computed, and the operation value is a null-value. Otherwise the right operand is computed and its value becomes the operation value.

The **&&** symbols can be used instead of the **and then** keywords.

The **or else** operation computes the left operand value first. If that value is not a null-value, the right operand is not computed, and the left operand value becomes the operation value. Otherwise the right operand is computed and its value becomes the operation value.

The **||** symbols can be used instead of the **or else** keywords.

The following expressions have the value of `"YES"`:

```
1+2 == 3 & 2+2 == 4
2+2 == 3 or else 7-5 == 2
false ^ true
```

The binary operations of the same priority group left-to-right: `X op Y op Z` is the same as `( X op Y ) op Z`

The binary operations have a higher priority than the ternary operations.

The language implements one ternary operation:

`cond ? expr1 : expr2`

- the *cond* operand is computed.
- If the computed value is not a null-value, the *expr1* operand is computed and its value becomes the operation value.
- Otherwise the *expr2* operand is computed and its value becomes the operation value.

The following expressions have the value of `"Good"`:

```
3 == 3 ? "Good" : "Bad"
null ? 77777 : "Good"
```

The ternary operation groups right-to-left: `A ? B : C ? D : E` is the same as `A ? B : (C ? D : E)`

The language implements the indexing operation:

`object[index]`

where

the *object* expression should have an array, a dictionary value, a string value, or a datablock (otherwise the operation results in a program exception), and

the *index* expression should have a numeric value smaller than the number of *object* elements, or the length of the *object* string or datablock.

If the *object* expression value is an array, the operation references the array element number *index* (the first element has the number 0).

Retrieving an element that does not exist results in a null-value.

Assigning a new value to the first array element that does not exist adds the value to the array as a new element.

An attempt to assign a new value to any other non-existing array element results in a program exception.

If SquareArray value is `(1,4,9,16,25)`, the following expression has the value of `9`:

`SquareArray[2]`

while the following operator:

```
        SquareArray[5] = "Blue";
```

changes the SquareArray value to `(1,4,9,16,25,"Blue")`,

If the *object* expression value is a dictionary, the operation references the dictionary key number *index* (the first key has the number 0).
Retrieving a key that does not exist results in a null-value.
An attempt to assign a new value to a referenced key results in a program exception.

If SquareDictionary value is `{"one" = 1; "two" = "four"; "three"=9;}`, the following expression has the value of `"three"`:

```
        SquareDictionary[2]
```

If the *object* expression value is a string, the operation returns a 1-byte string containing the string symbol number *index* (the first string symbol has the number 0).
Retrieving a symbol that does not exist results in a null-value.
An attempt to assign a new value to any string symbol results in a program exception.
If SquareString value is `"grass"`, the following expression has the value of `"r"`:

```
        SquareString[1]
```

If the *object* expression value is a datablock, the operation returns a number representing the datablock byte number *index* (the first datablock byte has the number 0).
Datablock bytes have unsigned numeric values in the 0..255 range.
Retrieving a byte that does not exist results in a null-value.
An attempt to assign a new value to any datablock byte results in a program exception.

The language implements the key, or dictionary-element operation:

```
        dictionary.keyName
```

where
        the *dictionary* expression should have a dictionary value (otherwise the operation results in a program exception), and
        the *keyName* is a name.

The operation references the *dictionary* element with the key *keyName*.
Retrieving an element for a key that does not exist results in a null-value.
Assigning a null-value to a key results in the key-value pair being removed from the dictionary.
Assigning a non-null-value to a key that does not exist results in a new key-value pair being added to the dictionary.
If SquareDictionary value is `{"one" = 1; "two" = "four"; "three"=9;}`,
the following expression has the value of `9`:

```
        SquareDictionary.three
```

and the following expression has the value of `null`:

```
        SquareDictionary.four
```

The language implements the computed key, or dictionary-element operation:

```
        dictionary.(keyExpr)
```

where
        the *dictionary* expression should have a dictionary value, and
        the *keyExpr* expression should have a string value (otherwise the operation results in a program exception).

The operation references the *dictionary* element for the *keyExpr* key.
If SquareDictionary value is `{"one" = 1; "two" = "four"; "three"=9;}`, the following expression has the value of `9`:

```
        SquareDictionary.("th" + "ree")
```

The language implements function calls. A function call is specified as a name followed by parentheses optionally containing a list of expressions (*parameter-expressions*):

```
functionName()
functionName(arg1)
functionName(arg1,arg2,...)
```

The *functionName* should be a name of a [built-in function](#) or an already [defined function](#).
Parameter expressions (if any) are computed, and the function code is executed using the parameter expression values.
A function result is an object, and the indexing and dictionary-element operations can be applied to a function result.

If MyFunction function has 2 parameters and its returned value is an array `(1,"four",9)`, the following expression has the value of `"four"`:

```
MyFunction(myData,"zzz")[1]
```

A function with one or more parameters can be called as a *method*:

```
arg1.functionName()
arg1.functionName(arg2)
arg1.functionName(arg2,arg3,...)
```

The MyFunction function call (see above) can be expressed as a method:

```
myData.MyFunction("zzz")[1]
```

# Operators

An operator sequence contains zero or more operators, followed by the semicolon (`;`) symbol.

An empty operator performs no action:

```
;
```

A null operator consists of the keyword `null`. It performs no action:

```
null;
```

An assignment operator consists of a data container reference (a variable, an array element, or a dictionary element), the assignment operation, and an expression. The expression and the data container reference are computed (in an unspecified order), and the expression value is used to modify the data container.
The = operation assigns the expression value to the data container.

```
myVar = 123 + 111;
myVar = "string";
```

Other assignment operations consist of a binary operation symbol and the = symbol. They apply the corresponding binary operation to the current data container value and the expression value, and store the result in the data container.

```
myVar[123] += 123; myVar.message += ", program stopped";
```

If a data container reference is an array element, that element must exist or it must be the first non-existent array element, i.e. if an array has 3 elements, you can assign values to elements number 0,1,2, and 3. In the last case, a new element is added to the array.

If a data container reference is a dictionary element, assigning a null-value effectively removes the element from the dictionary.

A procedure call operator is specified in the same way as a function call expression. The name used should be a name of a [built-in procedure](#) or an already [defined procedure](#).

If MyProc procedure has 2 parameters, the following operator is a correct procedure call:

```
MyProc(myData,"zzz");
```

For procedure with one or more parameters, a call operator can be specified as a *method*:

```
arg1.procedureName()
arg1.procedureName(arg2)
arg1.procedureName(arg2,arg3,...)
```

The MyProc call (see above) can be specified as a method:

```
myData.MyProc("zzz");
```

A stop operator terminates the Task execution. It consists of the **stop** keyword:

```
stop;
```

A return operator finishes function or procedure execution.
A return operator within a function consists of the **return** keyword and an expression. This expression is computed and its value becomes the value of the function call.

```
return 12*year+month;
```

A return operator within a procedure consists of the **return** keyword.

```
return;
```

A conditional operator consists of the **if** keyword followed by an expression (*if-expression*), the **end** keyword, an operator sequence (*if-sequence*), and the **end** keyword optionally followed by the **if** keyword.
The if-expression is computed, and if its value is not a null-value, the if-sequence is executed, and the conditional operation execution ends.

The following example increases the myCount variable value by 2 if its value is less than 10:

```
if myCount < 10 then
   myCount = myCount + 2;
end if;
```

A conditional operator can optionally contain one or more elif-portions after the if-sequence. Each elif-portion consists of the **elif** keyword, an expression (*elif-expression*), the **then** keyword, and an operator sequence (elif-sequence).
If the if-expression value is a null-value, the first elif-expression is computed, and if its value is not a null-value, its elif-sequence is executed and the conditional operation execution ends. If the elif-expression value is a null-value, the next elif-portion is processed.
In the following example the myCount variable value is checked. If the value is less than 10, the variable value is increased by 2. Otherwise (i.e. if the variable value is not less than 10), if the value is less than 20, it is decreased by 3:

```
if myCount < 10 then
   myCount = myCount + 2;
elif myCount < 20 then
   myCount = myCount - 3;
end if;
```

A conditional operator can optionally contain an else-portion. It is specified after the if-sequence and after all optional elif-portions. The else-portion consists of the **else** keyword and an operator sequence (*else-sequence*).
If the if-expression value is a null-value, and all optional elif-expression values are null-values, the else-sequence is executed and the conditional operator execution ends.
The following example increases the myCount variable value by 2 if the value is less than 10, it decreases the variable value by 3 if the value is not less than 10, but it is less than 20, and the variable value is multiplied by 4 in all other cases:

```
if myCount < 10 then
   myCount = myCount + 2;
elif myCount < 20 then
   myCount = myCount - 3;
else
   myCount = myCount * 4;
end if;
```

The loop operator consists of an optional preamble, the **loop** keyword, an operator sequence (*initial sequence*), and the **end** keyword optionally followed by the **loop** keyword.
Operators in the sequence are executed repeatedly.

A preamble for a *while-loop* is the **while** keyword and an expression (*while-expression*).
The while-expression is computed every time before the operator sequence is executed. If the while-expression result is a null-value, the loop operator ends.
A preamble for a *for-loop* is the **for** keyword optionally followed by an *initializer operator*, optionally followed by the **while** keyword and an expression (*while-expression*), optionally followed by the **by** keyword and an assignment operator (the *iterator operator*).
The *initializer operator* is either an assignment operator, or the **var** keyword followed by variable declarations. In the later case, these variables can be used only in the *while-expression*, *iterator operator*, and inside the loop operator sequence.
If the *initializer operator* is specified, it is executed once, when the loop operator starts.
If the *while-expression* is specified, it is computed every time before the loop operator sequence is executed. If the while-expression result is a null-value, the loop operator ends.
If the *iterator operator* is specified, it is executed every time after the loop operator sequence is executed.

The following example checks the myCount variable value and keeps increasing it by 2 while that value is less than 10:

```
while myCount < 10 loop
   myCount = myCount + 2;
end loop;
```

The following example calls the myProc procedure in an unconditional loop:

```
loop
   myProc(2,"test.wav");
end loop;
```

The following example calls the myProc procedure 3 times, with parameter values 5,7,9:

```
for i = 5 while i < 10 by i += 2 loop
   myProc(i);
end loop;
```

The loop operator can optionally contain one or more exitif-portions between the initial sequence and the **end** keyword. Each exitif-portion consists of the **exitif** keyword, followed by an expression (exitif-expression), the semicolon (`';'`) symbol, and an operator sequence. After the initial sequence is executed, the first exitif-expression is computed. If its value is not null, the loop operator execution ends. Otherwise the exitif operator sequence is executed, and the next exitif expression is computed. After the last exitif operator sequence is executed, the loop operator execution repeats.

The following example appends the `"aaa"` string to the myWord variable, checks the variable lengths, and if the length is less than 20, appends the `"bbb"` string to the myWord variable, and repeats the process:

```
loop
   myWord += "aaa";
exitif length(myWord) >= 20;
   myWord += "bbb";
end loop;
```

## Alternative Forms

Some operators in a sequence may be presented in an *alternative* form. Operators in an *alternative* form are not followed by the semicolon symbol.

The alternative form of a conditional operator consists of the **if** keyword followed by an expression (*if-expression*), and an operator sequence (*if-sequence*) enclosed in left (`{`) and right (`}`) brace symbols.
The alternative form of a conditional operator can optionally contain one or more elif-portions after the enclosed if-sequence. Each elif-portion consists of the **elif** keyword, an expression (*elif-expression*), and an operator sequence (elif-sequence), enclosed into braces.
The alternative form of a conditional operator can optionally contain an else-portion. It is specified after the enclosed if-sequence and after all optional elif-portions. The else-portion consists of the **else** keyword and an operator sequence (*else-sequence*) enclosed in braces.

The following example increases the myCount variable value by 2 if the value is less than 10, it decreases the variable value by 3 if the value is not less than 10, but it is less than 20, and the variable value is multiplied by 4 in all other cases:

```
if (myCount < 10) {
```

```
   myCount = myCount + 2;
} elif (myCount < 20) {
   myCount = myCount - 3;
} else {
   myCount = myCount * 4;
}
```

**Note:** it is not required to use parentheses to enclose if-expressions or elif-expressions.

The alternative form of a while-loop operator consists of the **while** keyword, an expression (*while-expression*), the left brace ('{') symbol, an operator sequence (*initial sequence*), zero or more exitif-portions and a right brace ('}') symbol.

The alternative form of a for-loop operator consists of the **for** keyword, the left parenthesis ('(') symbol, an optional *initializer operator*, the semicolon (';') symbol, an optional *while-expression*, the semicolon (';') symbol, an optional *iterator operator*, the right parenthesis (')') symbol, the left brace ('{') symbol, an operator sequence (*initial sequence*), zero or more exitif-portions and a right brace ('}') symbol.

Each optional exitif-portion consists of the **exitif** keyword, followed by an expression (exitif-expression), the semicolon (';') symbol, and an operator sequence.

The following example appends the `"aaa"` string to the myWord variable, checks the variable lengths, and if the length is less than 20, appends the `"bbb"` string to the myWord variable, and repeats the process:

```
while ( true ) {
   myWord = myWord + "aaa";
exitif (length(myWord) >= 20);
   myWord = myWord + "bbb";
}
```

**Note:** it is not required to use parentheses to enclose while-expressions or exitif-expressions.

The following example assigns the factorial of the variable X value to the myFactor variable.

```
myFactor = 1; for(i = 0; i <= X; i += 1) {
   myFactor *= i;
}
```

# Built-in Procedures and Functions

The following procedures and functions are built into the language, and they are available to all applications. You should not use their names for user-defined procedures or functions.

Same(*arg1,arg2*)

>This function returns a true-value if the values of *arg1* and *arg2* are the same object or if both values are null-values or both values are true-values. In other cases the function returns a null-value.
>
>The value of Same("J" + "ack","Jack") is a null-value.
>
>In the following example:
>
>```
>x = "my string";
>y = "my string";
>z = x;
>test1 = Same(x,y);
>test2 = Same(x,x);
>test3 = Same(x,z);
>```
>
>the resulting value of test1 is a null-value, while the values of test2 and test3 are true-values.

Copy(*arg*)

>If the *arg* value is a null-value, this function returns a null-value.
>
>Otherwise, the function returns the copy of the *arg* value.
>
>For complex objects (such as arrays, dictionaries, XML objects), this function copies all complex object elements, too.

Length(*arg*)

>If *arg* is a string, this function returns the string size (in bytes);
>
>If *arg* is a datablock, this function returns the datablock size (in bytes);
>
>If *arg* is an array, this function returns the number of array elements;
>
>If *arg* is a dictionary, this function returns the number of dictionary keys;
>
>If *arg* is a Mailbox handle, this function returns the number of messages in the Mailbox;
>
>In all other cases, this function returns the number 0.

```
Min(arg1,arg2)
```
If the *arg1* value < the *arg2* value, then the function returns the *arg1* value, otherwise it returns the *arg2* value.
See above for the < operation definition.

```
Max(arg1,arg2)
```
If the *arg1* value > the *arg2* value, then the function returns the *arg1* value, otherwise it returns the *arg2* value.
See above for the > operation definition.

```
Void(arg1)
```
This procedure does nothing, it just discards the *arg1* value. Use this procedure when you need to call a function, but you do not want to use the function result.

```
ProcCall(name)
ProcCall(name,arg1)
ProcCall(name,arg1,arg2)
ProcCall(name,arg1,....)
```
This procedure has 1 or more parameters. The *name* value should be a string, it specifies a simple or a qualified external procedure name.
This external procedure is called, using the remaining parameters as its parameters.
There is no need to use external-declarations for procedures called this way.

```
ProcCall("myModule::myProc",123,"2222");
```
is the same as
```
procedure myModule::myProc(x,y) external;
myModule::myProc(123,"2222");
```

```
FuncCall(name)
FuncCall(name,arg1)
FuncCall(name,arg1,arg2)
FuncCall(name,arg1,....)
```
This function has 1 or more parameters. The *name* value should be a string, it specifies a simple or a qualified external function name.
This external function is called, using the remaining parameters as its parameters.
There is no need to use external-declarations for functions called this way.

```
x = FuncCall("myModule::myFunc",123,"2222");
```
is the same as
```
function myModule::myFunc(x,y) external;
x = myModule::myFunc(123,"2222");
```

```
objectClass(arg)
```
the function returns a string with the name of the class the *arg* value belongs to ("STString" for strings, "STNumber" for numbers, "STDate" for timestamps, "STData" for datablocks, etc.)

## Strings

```
IsString(arg)
```
This function returns a true-value if the *arg* value is a string, otherwise the function returns a null-value.

```
String(arg)
```
If the *arg* value is a string, this function returns this string.
If the *arg* value is a number, this function returns a string with the decimal representation of that number.
If the *arg* value is a datablock, this function returns a string with the datablock content (interpreted as textual data).
If the *arg* value is a timestamp, this function returns a string with the system-standard timestamp text representation.
If the *arg* value is an ip-address, this function returns a string with the canonical representation of that network address.
If the *arg* value is an XML Object, this function returns the XML object text body ("text node").
If the *arg* value is a null-value, this function returns a null-value.
In all other cases, this function returns a string with a textual representation of the *arg* value.

```
Substring(str,from,len)
```
If the *str* value is a string, and the *from* value is a number, and the *len* value is a non-negative number, this function returns a string that is a substring of the *str* value, with the *len* length.
If *from* has a non-negative value, the substring starts at the *from* position (the first symbol of the string has the position 0).
If *from* is a negative value, then the substring ends at the -1-*from* position from the string end (to include the last *str* symbol(s), *from* should be -1).

If the *from* value (or -1-*from* value) is equal to or greater than the *str* value length, the result is an empty string.

If the *from + len* (or -1-*from + len*) value is greater than the *str* value length, the resulting string is shorter than *len*.

In all other cases this function returns a null-value.

**Note:** this function interprets a string as a sequence of bytes, and thus it can break non-ASCII symbols which are stored as a multi-byte sequence.

```
FindSubstring(str,substr)
FindSubstring(str,substr,offset)
```
The *str* and *substr* values should be strings, while the *offset* should be a number. The function checks if the *substr* value is a substring of the *str* value.

If the *offset* parameter is specified and its value is greater than zero, then the search ignores the first *offset* bytes of the *str* string value, and the position of the first found substring in the *str* value is returned.

If the *offset* parameter is specified and its value is less than zero, then the search ignores the last -1-*offset* bytes of the *str* string value, and the position of the last found substring in the *str* value is returned.

All positions start with 0.

If no substring is found, this function returns the number -1.

Example 1. The value of `FindSubstring("forest","for")` is 0.

Example 2. The value of `FindSubstring("A forest","for")` is 2.

Example 3. The value of `FindSubstring("A forest for all","for",1)` is 2.

Example 4. The value of `FindSubstring("A forest for all","for",4)` is 9.

Example 5. The value of `FindSubstring("A forest for all","for",-3)` is 9.

Example 6. The value of `FindSubstring("A forest for all","for",-5)` is 2.

```
Range(str,from,len)
```
This function works in the same way as the `Substring` function, but if the *str* value is a string, it is interpreted as a sequence of "glyphs" - single and multi-byte sequences representing ASCII and non-ASCII symbols. The *from* and *len* values specify the substring position and length in terms of symbols, rather than bytes.

If the *str* value is a datablock, the function returns a datablock containing a portion of the *str* value, cut using the same rules, while the *from* and *len* values specify the position position and length in bytes.

```
EOL()
```
This function returns a string containing the EOL (end-of-line) symbol(s) used on the Server OS platform.

```
CRLF()
```
This function returns a string with the Internet EOL (<carriage-return><line-feed>) symbols.

```
ToUpperCase(str)
```
If the *str* value is a string, the function result is this string with all its letters converted to the upper case.

```
ToLowerCase(str)
```
If the *str* value is a string, the function result is this string with all its letters converted to the lower case.

```
IsDigit(str)
```
This function returns a true-value if the *str* value is a 1-symbol string and that symbol is a decimal digit (0..9), otherwise the function returns a null-value.

```
IsWhiteSpaces(str)
```
This function returns a true-value if the *str* value is a string containing only space, <tab>, <carriage-return> or <line-feed> symbols, otherwise the function returns a null-value.

```
FindRegEx(str,picture)
```
The function compares the *str* string with the *picture* string containing a *regular expression*.

If *str* is not a string, or *picture* is not a string, or the *picture* string cannot be interpreted as a *regular expression*, or the *str* string does not match the *regular expression*, the function returns a null-value.

Otherwise, the function returns an array of strings. Its zero element contains a copy of the *str* string, while additional elements (if any) contain the *str* substrings matching the *regular expression groups*.

The *picture* string should specify a *regular expression* using the *extended POSIX syntax*.

```
EmailDomainPart(address)
```
If the *address* value is a string, and the string contains the @ symbol, this function returns a string containing the *address* string part after its first the @ symbol.

Otherwise, the function returns a null-value.

```
EmailUserPart(address)
```

If the *address* value is not string, this function returns a null-value.

If the *address* value is a string not containing the `@` symbol, this function returns the same string.

Otherwise (the *address* value is a string containing the `@` symbol), this function returns the *address* string part before the first `@` symbol.

## Numbers

`IsNumber(`*arg*`)`
> This function returns a true-value if the *arg* value is a number, otherwise the function returns a null-value.

`Number(`*arg*`)`
> If the *arg* value is a number, this function returns this number.
> If the *arg* value is a string, this function returns the numeric value of that string, till the first non-numeric symbol.
> For example, the value of `Number("123#")` is `123`.
> In all other cases, this function returns the number `0`.

`RandomNumber()`
> This function returns a random integer number in the `[0..9223372036854775807]` (`[0..2*63-1]`) range.

## TimeStamps

`IsDate(`*arg*`)`
> This function returns a true-value if the *arg* value is a timestamp, otherwise the function returns a null-value.

`Date(`*arg*`)`
> If the *arg* value is a timestamp, this function returns this timestamp.
> If the *arg* value is the number `0`, this function returns the "remote past" timestamp.
> If the *arg* value is a negative number, this function returns the "remote future" timestamp.
> If the *arg* value is a positive number N, this function returns the timestamp corresponding to the start of the Nth day, where the 1st day is 2nd of January, 1970.
> If the *arg* value is a string, if should be a textual representation of some timestamp, and this function returns that timestamp.
>
> In all other cases, this function returns a null-value.

`GMTTime()`
> This function returns a timestamp object with the current GMT time.

`LocalTime()`
> This function returns a timestamp object with the current local time.

`GMTToLocal(`*arg*`)`
> If the *arg* value is a timestamp object, this function returns a timestamp object containing the *arg* value converted from the GMT to the local time.
> In all other cases, this function returns a null-value.

`LocalToGMT(`*arg*`)`
> If the *arg* value is a timestamp object, this function returns a timestamp object containing the *arg* value converted from the local time to the GMT.
> In all other cases, this function returns a null-value.

`Year(`*arg*`)`
> If the *arg* value is a timestamp object, this function returns a number containing the *arg* value year.
> If the the timestamp value is the "remote past" timestamp, the function returns the number `0`.
> If the the timestamp value is the "remote future" timestamp, the function returns the number `9999`.
> In all other cases, this function returns a null-value.

`Month(`*arg*`)`
> If the *arg* value is a timestamp object, this function returns a string containing the *arg* value month name (`Jan`, `Feb`, `Mar`, `Apr`, `May`, `Jun`, `Jul`, `Aug`, `Sep`, `Oct`, `Nov`, `Dec`).
> If the *arg* value is a number in the `1..12` range, this function returns a string containing the name of month number *arg* (`Month(1)` returns `Jan`).

In all other cases, this function returns a null-value.

MonthNum(*arg*)

If the *arg* value is a timestamp object, this function returns the *arg* value month number (`1` for January).

In all other cases, this function returns a null-value.

MonthDay(*arg*)

If the *arg* value is a timestamp object, this function returns a number containing the day of month for the *arg* value date (`1` is returned for the first day of month).

In all other cases, this function returns a null-value.

WeekDay(*arg*)

If the *arg* value is a timestamp object, this function returns a string containing the name of week day of the *arg* value date (`Mon`, `Tue`, `Wed`, `Thu`, `Fri`, `Sat`, `Sun`).

In all other cases, this function returns a null-value.

YearDay(*arg*)

If the *arg* value is a timestamp object, this function returns a number containing the day of year for the *arg* value date (the number 1 is returned for the 1st of January).

In all other cases, this function returns a null-value.

TimeOfDay(*arg*)

If the *arg* value is a timestamp object, this function returns the number of seconds between the date *arg* value and the start of its day.

In all other cases, this function returns a null-value.

DateNumber(*arg*)

If the *arg* value is a timestamp object, this function returns the number of full days between the *arg* value date and 01-Jan-1970.

In all other cases, this function returns a null-value.

DateByMonthDay(*year*,*monthNum*,*monthDay*)

The *year*, *monthNum*, *monthDay* values should be positive numbers. If any of these values is incorrect, this function returns a null-value. Otherwise, the function returns a timestamp object presenting midnight of the specified date.

The following expression timestamp value is midnight, 05-Nov-2008:

```
DateByMonthDay(2008,11,5)
```

DateByYearDay(*year*,*yearDay*)

The *year*, *yearDay* values should be positive numbers. If any of these values is incorrect, this function returns a null-value. Otherwise, the function returns a timestamp object presenting midnight of the specified date.

The following expression timestamp value is midnight, 01-Feb-2006:

```
DateByYearDay(2006,32)
```

## IP Addresses

IsIPAddress(*arg*)

This function returns a true-value if the *arg* value is an ip-address, otherwise the function returns a null-value.

IPAddress(*arg*)

If the *arg* value is an ip-address, this function returns this ip-address.

If the *arg* value is a string with a correct presentation of an IP address (with an optional port number), the function returns that ip-address.

In all other cases, this function returns a null-value.

## Datablocks

IsData(*arg*)

This function returns a true-value if the *arg* value is a datablock, otherwise the function returns a null-value.

```
RandomData(length)
```
The *length* value should be a positive numbers not larger than 4096.

This function returns a datablock of the specified length, filled with random data.

```
EmptyData()
```
This function returns an empty datablock.

# Arrays

```
IsArray(arg)
```
This function returns a true-value if the *arg* value is an array, otherwise the function returns a null-value.

```
NewArray()
```
This function returns a newly created empty array.

```
Invert(arg)
```
The *arg* value should be an array, otherwise this function call results in a program exception.

This function returns an array containing the same elements as the *arg* value, but in the reversed order.

```
Find(source,object)
```
If the *source* value is an array, this function returns a number - the index in the array for the first element equal to the *object* value. If the *source* array does not contain such an object, a negative numeric value is returned.

If the *source* value is a [Calendar handle](), calendar items are returned (see below).

If the *source* value is anything else, a negative numeric value is returned.

```
AddElement(target,element)
```
If the *target* value is an array, this procedure adds *element* value as the new last element of that array.

If the myArray value is `(1,4,9,16,25)`, the `AddElement(myArray,"test")` call changes the myArray value to `(1,4,9,16,25,"test")`.

If the *target* value is an XML Object, this procedure adds *element* value as its new sub-element (the *element* value should be a string or an XML Object).

In all other cases this procedure call results in a program exception.

```
RemoveElement(target,what)
```
This procedure removes an element from an array.

If *target* value is an array, the *what* value should be a number or a string containing a decimal number, specifying the array element to remove.

If the myArray value is `(1,4,9,16,25)`, the `RemoveElement(myArray,2)` call changes the myArray value to `(1,4,16,25)`.

In all other cases this procedure call results in a program exception.

```
InsertElement(target,index,element)
```
This procedure inserts an element into an array.

The *target* value should be an array, otherwise this procedure call results in a program exception.

The *index* value should be a number or a string containing a decimal number, specifying where to insert the *element* value.

All existing array elements with index number of *index* and bigger increase their index number by one.

If the myArray value is `(1,4,9,16,25)`, the `InsertElement(myArray,2,"Jack")` call changes the myArray value to `(1,4,"Jack",9,16,25)`.

```
SortStrings(arg)
```
This procedure sorts array elements.

The *arg* value should be an array, and all array elements must be strings, otherwise this procedure call results in a program exception.

The array elements are compared as case-insensitive UTF-8 strings.

# Dictionaries

```
IsDictionary(arg)
```
This function returns a true-value if the *arg* value is a dictionary, otherwise the function returns a null-value.

```
NewDictionary()
```
This function returns a newly created empty dictionary.

```
SetCaseSensitive(target,flag)
```
This procedure specifies if dictionary keys should be processes as case-sensitive.

The *target* value should be a dictionary, otherwise this procedure call results in a program exception.

If *flag* value is a null-value, the *target* dictionary becomes case-insensitive, otherwise it becomes case-sensitive.

New dictionaries are created as case-sensitive.

# XML Objects

```
IsXML(arg)
```
This function returns a true-value if the *arg* value is an XML object, otherwise the function returns a null-value.

```
NewXML(type)
NewXML(type,prefix)
NewXML(type,prefix,namespace)
```
This function returns a newly created XML Object of type *type*. The *type* value should be a string containing a valid XML tag.

The *prefix* value (if specified) can be a null-value or a string containing a valid XML prefix. This prefix is used to qualify the XML object name, if not specified or if its value is a null-value, then an empty string value is used.

If the *namespace* value is specified and its value is not a null-value, it should be a string containing the namespace to be associated with the *prefix* value.

```
SetNamespace(data,namespace)
SetNamespace(data,namespace,prefix)
```
This procedure associates an XML prefix with some namespace.

The *data* value should be an XML Object, otherwise this procedure call results in a program exception.

If the *prefix* is not specified, or its value is a null-value, then an empty string value is used. Otherwise, its value should be a string containing a valid XML prefix.

If the *namespace* value is a null-value, the namespaces associated with the *prefix* value is removed. Otherwise the *namespace* value should be a string, containing the namespace to be associated with the *prefix* value.

```
GetNamespace(data)
GetNamespace(data,prefix)
```
This function returns a string with the namespace associated with the specified XML prefix, or a null-value if there is no associated namespace.

The *data* value should be an XML Object, otherwise this function call results in a program exception.

If the *prefix* is not specified, or its value is a null-value, then an empty string value is used. Otherwise, its value should be a string containing a valid XML prefix.

```
GetPrefix(data,namespace)
```
This function returns a string with the prefix assigned to the specified namespace, or a null-value if the namespace is not included into the XML object.

The *data* value should be an XML Object, otherwise this function call results in a program exception.

The *prefix* value should be a string containing a valid XML prefix.

```
SetAttribute(data,value,name)
SetAttribute(data,value,name,prefix )
```
This procedure sets an XML Object attribute for the specified name and prefix.

The *data* value should be an XML Object, otherwise this procedure call results in a program exception.

The *name* value should be a string containing a valid XML attribute name.

If the *prefix* parameter is specified, its value should be a null-value or a string containing a valid XML prefix.

If the *value* value is a null-value, the attribute with the specified name and prefix is removed. Otherwise the *value* value should be a string containing the new attribute value.

```
GetAttribute(data,name)
GetAttribute(data,name,prefix)
```

The *data* value should be an XML Object, otherwise this function call results in a program exception.

If the *prefix* parameter is specified, its value value should be a null-value or a string containing a valid XML prefix.

If the *name* value is a string, it should contain a valid XML attribute name. In this case this function returns a string value of the attribute with the specified name and prefix, or a null-value if there is no such attribute.

If the *name* value is a number, it should specify the attribute index (starting from 0). If an attribute with this index does not exist, this function returns a null-value. Otherwise, this function returns an array. The first array element is the attribute name, the second array element is the attribute prefix.

If the *name* value is not a string and it is not a number, this function call results in a program exception.

```
XMLBody(data,type)
XMLBody(data,type,namespace)
XMLBody(data,type,namespace,index)
```
This function returns an XML sub-element with the specified type, namespace, and position, or a null-value if there is no such sub-element.

The *data* value should be an XML Object, otherwise this function call results in a program exception.

The *type* value should be a null-value or a string containing a valid XML tag name. If the *type* value is a null-value, the function retrieves sub-elements of any type.

If the *namespace* parameter is specified, its value should be a null-value or a string containing a namespace. If the *namespace* is not specified, or its value is a null-value, the function looks for sub-elements ignoring their namespaces.

If the *index* parameter is specified, its value should be a non-negative number. To retrieve the first sub-element with the specified type and namespace, the *index* value (if specified) should be `0`.

```
XMLType(data)
```
This function returns a string with the XML Object type.

The *data* value should be an XML Object, otherwise this function call results in a program exception.

```
XMLPrefix(data)
```
This function returns a string with the XML Object type prefix.

The *data* value should be an XML Object, otherwise this function call results in a program exception.

# Data Conversion

```
ObjectToString(arg)
```
This function returns a string with a textual representation of the *arg* value.

If the *arg* value is a null-value, the function returns the `"#null#"` string.

```
ObjectToXML(arg)
```
This function returns an XML representation of the *arg* value, as specified in [XML Objects](#) section.

```
ToObject(arg)
```
If the *arg* value is a string or a datablock, this function returns an object textually represented with the *arg* value.

If the *arg* value is an XML object, this function returns an object this XML represents (as specified in [XML Objects](#) section).

Otherwise this function call results in a program exception.

If conversion fails or if the *arg* value is the `"#null#"` string, the function returns a null-value.

```
Base64Encode(arg)
```
The *arg* value should be a string or a datablock, otherwise this function call results in a program exception.

This function returns a string containing the base64-encoded *arg* data.

```
Base64Decode(arg)
```
The *arg* value should be a string, otherwise this function call results in a program exception.

This function returns a datablock containing the base64-decoded *arg* data.

```
AppToXML(data,format)
```
This function converts application data into its XML representation.

The *data* value should be a string or a datablock containing the application data text.

The *format* value should be a string specifying the application data format. The following formats are supported:

- `vCard` - vCard format, the function returns either a [vCard](#) XML object, or an array of vCard XML objects.
- `vCardGroup` - vCardGroup format, the function returns a [vCardGroup](#) XML object.
- `iCalendar` - iCalendar format, the function returns an [iCalendar](#) XML object.
- `sdp` - SDP format, the function returns an [SDP](#) XML object.

If the function fails to parse the application data, or the specified format is not supported, the function returns an error code string.

```
XMLToApp(data)
```
This function converts an XML representation into application data (see above).

The *data* value should be an XML object.

The function returns the application data string.

If the XML object is not a representation of some supported application data format, the resulting string is a generic textual representation of the *data* value XML object.

If the function fails to convert the XML representation into application data, the resulting string contains the error code.

ObjectToJSON(*arg*)

This function returns a string with a JSON (JavaScript Object Notation) representation of the *arg* value.

Null-values are represented as `null` tokens.

Timestamp objects are represented as RFC822-formatted date strings.

JSONToObject(*arg*)

If the *arg* value is a string or a datablock, this function returns an object textually represented with the *arg* value, using the JSON format.

Otherwise this function call results in a program exception.

If conversion fails or if the *arg* value is the `null` string, the function returns a null-value.

Convert(*data*,*format*)

This function converts the `data` value into the format specified with the *format* value, which should be a string.

The following *format* values are supported (case-insensitive, unless explicitly specified):

"base64"

If the *data* value is a string, the function returns a datablock with its base64-decoded data.

If the *data* value is a datablock, the function returns a string with its base64-encoded data.

"hex"

If the *data* value is a string, it should contain hexadecimal symbols, and the function returns a datablock with its decoded data.

If the *data* value is a datablock, the function returns a string with its hexadecimal encoding. The `"HEX"` format produces upper-case encodings, the `"hex"` format produces lower-case encodings.

"ucs-2[+][le|be]"

If the *data* value is a string, the function returns a datablock with its ucs-2 encoded data.

If the *data* value is a datablock, the function returns a string with its ucs-2 decoded data.

The `le` or `be` suffix specifies the ucs-2 encoding byte order (little-endian or big-endian, respectively). The big-endian byte order is used by default.

If the datablock to be decoded contains a BOM (byte order mark), the byte order suffix is ignored.

The `+` symbol causes the BOM (byte order mark) to be placed into the encoded data. This symbol is ignored when decoding.

"*charsetName*" (any character set name known to the CommuniGate Pro Server)

If the *data* value is a string, the function returns a datablock with its data encoded using the specified character set.

If the *data* value is a datablock, the function returns a string with its data decoded from the specified character set.

"utfcode"

If the *data* value is a string, the function returns a number - the Unicode code of the first string symbol.

If the *data* value is a number, it is interpreted as a Unicode code, and the function returns a string consisting of this symbol.

"words"

If the *data* value is a string, the function splits it into words - non-empty substrings separated with sequences of "white space" and "end of line" symbols, and returns an array containing all "word" strings.

"asn.1"

If the *data* value is an array, it should represent some ASN.1 data structure. The function returns a datablock with ASN.1 BER encoding of this structure.

If encoding fails, the function returns an error code string.

If the *data* value is a datablock, the function returns an array representing this ASN.1 data structure.

If decoding fails, the function returns an error code string.

"dns-a"

If the *data* value is a string, the function peforms a DNS A-record search for this domain name.

The function returns either an array of IP Addresses from the found records, or an error code string.

`"dns-aaaa"`
> Same as "dns-a", but for the DNS AAAA-record.

`"dns-ptr"`
> If the *data* value is a string, the function peforms a DNS PTR-record search for this domain name.
> If the *data* value is an IP Address, the function peforms a DNS "IN-Addr" PTR-record search for that address.
> The function returns either an array with one string element - the found domain name, or an error code string.

`"dns-txt"`
> If the *data* value is a string, the function peforms a DNS TXT-record search for this domain name.
> The function returns either an array containing the found record strings, or an error code string.

`"gennonce"`
> The function returns a datablock with a "nonce" data. The *data* value should be a number, it specifies the "nonce" size in bytes. If this value is outside the supported nonce size range, the function returns a null-value.

`"checknonce"`
> If the *data* value is a datablock, the function returns a true-value if this datablock is a valid "nonce", otherwise the function returns a null-value.

`SystemInfo(`*`what`*`)`
> This function retrieves platform-specific data specified with the *what* value, which should be a string.
> The following *what* values are supported (case-insensitive, unless explicitly specified):

`"serverVersion"`
> The function returns a string with this CommuniGate Pro Server version.

`"serverOS"`
> The function returns a string with the name of the Operating System controlling the computer this CommuniGate Pro Server is running on.

`"serverCPU"`
> The function returns a string with the type of processor(s) this CommuniGate Pro Server is running on.

`"mainDomainName"`
> The function returns a string with the CommuniGate Pro Server Main Domain Name.

`"licenseDomainName"`
> The function returns a string with the Cluster Domain Name for a Cluster system, the Main Domain Name or a non-clustered system.

`"startTime"`
> The function returns a timestamp value specifying the time when this CommuniGate Pro Server instance started.

`"serverInstance"`
> The function returns a string which is unique for each running instance of the CommuniGate Pro Server.

`"clusterInstance"`
> The function returns a string which is unique for each running instance of the CommuniGate Pro Cluster, and it has the same value for all cluster members.

# Cryptography

`CryDigest(`*`algName,data`*`)`
`CryDigest(`*`algName,data,salt`*`)`
> This function computes the cryptographic digest of the *data* datablock.
> The *algName* value should be a string specifying the digest algorithm name (`MD5`, `SHA1`, etc).
> If the third parameter *salt* is specified, the digest is calculated using the `HMAC` algorithm with this third parameter used as the key and *algName* used for both HMAC stages.
> If the algorithm with the specified name is unknown, the function returns a null-value, otherwise the function returns a

datablock with the calculated digest.

`CryEncrypt(`*`algName,key,data`*`)`
> This function encrypts the *data* datablock using the *key* datablock as the encryption key.
>
> The *algName* value should be a string specifying the cipher algorithm name (`RC4`, `DES3`, `AES`, etc).
>
> If the algorithm with the specified name is unknown, the function returns a null-value, otherwise the function returns a datablock with the encrypted data.

`CryDecrypt(`*`algName,key,data`*`)`
> This function decrypts the *data* datablock using the *key* datablock as the decryption key.
>
> The *algName* value should be a string specifying the cipher algorithm name.
>
> If the algorithm with the specified name is unknown, the function returns a null-value, otherwise the function returns a datablock with the decrypted data.

# Environments

`Vars()`
> This function returns a dictionary unique for this *Task* (a program invocation). This dictionary can be used to store "task variables" visible in all procedures and functions executed in this Task.
>
> When a Task is started with parameters (for example, when a [Real-Time Application](#) is started to process a Signal directed with the [Router](#)), the parameter array is placed into the `startParameter` element of the `Vars()` dictionary.
>
> The following example retrieves the first two Task parameters:

```
firstParam  = Vars().startParameter[0];
secondParam = Vars().startParameter[1];
```

# Addresses and URIs

`SIPURIToEmail(`*`uri`*`)`
> This function converts the *uri* value from a SIP URI into an E-mail string.
>
> If the *uri* value is not a string, or if it cannot be parsed as a SIP URI, the function returns a null-value.

`EmailToSIPURI(`*`email`*`)`
> This function converts the *email* value from an E-mail address into a SIP URI string.
>
> If the *email* value is not a string, or if it cannot be parsed as an E-mail, the function returns a null-value.

`PhoneNumberToSIPURI(`*`phoneNumber`*`)`
> This function converts the *phoneNumber* value into a SIP URI string.
>
> If the *email* value is not a string, or if it cannot be parsed as a telephone number, the function returns a null-value.
>
> The function removes all formatting symbols from the *phoneNumber* value, leaving only the digits and the leading plus (`+`) symbol (if it exists).
>
> The function adds the current Domain name to the converted number.

`RouteAddress(`*`email,type`*`)`
> This function uses the [Router](#) to process the *email* address.
>
> The *type* value specifies the address type: it should be the `mail`, `signal`, or `access` string.
>
> If address routing fails, the function returns an error code string. Otherwise, the function result is a dictionary with the following elements:

> `module`
>> the name of the CommuniGate Pro module that will process this address.

> `host`
>> a string with the name of the host (a local Account, a remote domain, etc.) the address is routed to.

> `object`
>> a string with the name of the host object the address is routed to.

> `canRelay`
>> this optional element exists and contains a true-value if information can be relayed to this address.

RouteENUM(`service`,`phoneNumber`,`domain`)
> This function uses the [Domain Name Resolver](#) to convert the *phoneNumber* telephone number (a string with any sequence of digits with an optional leading `+` symbol) into a URL.
> The *service* value should be a string. It specifies the ENUM "service" (such as `"E2U+SIP"` or `"E2U+MAIL"`).
> The *domain* value should be a string. It specifies the ENUM domain name (such as `"e164.arpa"`).
> If the telephone number has been converted successfully, this function returns an array. Its first element contains a string - the resulting URL.
> Otherwise, this function returns an error code string.

# Account Data

The following functions and procedures are available when the program (a Task) has a "current Account" set.

MyDomain()
> This function returns a string with the current Domain name, if there is one. If there is no Account or Domain associated with the current Task, this function returns a null-value.

MyEmail()
> This function returns a string with the current Account E-mail, if there is one. If there is no Account associated with the current Task, this function returns a null-value.

GetAccountPreferences(`keyName`)
> This function returns Preference data for the current Account.
> If the *keyName* is a non-empty string, the Account Preference object for that key is returned. Otherwise a dictionary with all effective Account Preferences is returned.
> If the *keyName* string starts with a `~username/` prefix, the prefix is removed. The *username* string specifies the name of the Account to use. If this Account is not the same as the current Account, the operation succeeds only if the current Account has a Domain Administrator right for the specified Account Domain.

SetAccountPreferences(`keyValue`,`keyName`)
> This function updates Preference data for the current Account.
> If the *keyName* is a non-empty string, the *keyValue* value specifies the new object for that key. If the *keyValue* is a null-value, the object is removed from the Account Preference data, enabling the default value for the specified key.
> If the *keyName* string starts with a `~username/` prefix, the prefix is removed. The *username* string specifies the name of the Account to update. If this Account is not the same as the current Account, the operation succeeds only if the current Account has a Domain Administrator right for the specified Account Domain.
> If the *keyName* is a `~username/` string, or an empty string, or if it is not a string, the *keyValue* value must be a dictionary. It is used to update the Account Preference Data.
> This function returns a null-value if Preference data is successfully updated, otherwise it returns a string with an error code.

Impersonate(`email`)
> This function Routes the *email* address and sets the result as the current Account.
> If the routed address is not local, the current Account is cleared.
> If the routed address is local and it is not the same as the current Account, the current Account must have the [CanImpersonate](#) access right for the routed address Domain.
> When the current Account is changed, the preferences, the selected language, and the selected time zone are changed, too.
> This function returns a null-value if the operation has succeeded, otherwise it returns a string with an error code.

ReadGroupMembers(`groupName`)
> This function reads a [Group](#).
> The *groupName* value should be a string. It specifies the name of the Group to read. If this name does not contain a Domain name, the current Domain is used.
> This function returns an array of strings containing Group member E-mail addresses. This function returns a null-value if there is no Group with the specified name.
> The current Account should have the Domain Administrator right for the Group Domain.

ReadTelnums()
ReadTelnums(`accountName`)
> This function reads [Phone Numbers](#) assigned to the *accountName* Account or to the current Account, if the *accountName* parameter is not specified or if its value is a null-value.

This function returns an array of strings. This function returns a null-value if there is no Account with the specified name.
To retrieve the Phone Numbers assigned to a different Account, the current Account should have the Domain Administrator right for the *accountName* Account Domain.

```
UpdateAccountMailRule(ruleData)
UpdateAccountMailRule(ruleData,accountName)
UpdateAccountSignalRule(ruleData)
UpdateAccountSignalRule(ruleData,accountName)
```
These functions modify Account Queue or Signal Rules.

The *ruleData* parameter is a string or an array. It has the same meaning as the *newRule* parameter of the UpdateAccountMailRule and UpdateAccountSignalRule CLI commands.

These functions update the Rules of the the *accountName* Account, or the current Account if the *accountName* parameter is not specified or its value is a null-value.

To update Queue Rules of a different Account, the current Account should have the RulesAllowed Domain Administrator right for the *accountName* Account Domain.

To update Signal Rules of a different Account, the current Account should have the SignalRulesAllowed Domain Administrator right for the *accountName* Account Domain.

This function returns a null-value if the operation has succeeded, otherwise it returns a string with an error code.

## Mailboxes

```
ListMailboxes(filter)
```
This function lists all Mailboxes in the current Account.

The *filter* value should be a string - it specifies the search pattern. If the value is a null-value, the search pattern `*` (all Account Mailboxes) is used.

To search Mailboxes in a different Account, the search pattern should be specified as a `"~accountName/pattern"` string.

If the operation is successful, this function returns a dictionary. Each dictionary key is a string with the found Mailbox name.

The dictionary element value is:

- a dictionary with the Mailbox attributes - if the found Mailbox is an "item container" only.
- an empty array - if the found Mailbox is a "mailbox folder" only.
- an array with one dictionary element- if the found Mailbox is both a "mailbox folder" and an "item container".

If the Mailbox Access rights allow the current Account to see that Mailbox, but do not allow the current Account to open it, the Mailbox Attribute dictionary is replaced with a string containing the Mailbox Access Rights for the current Account.

If this function fails, it returns a string with an error code.

```
CreateMailbox(mailboxName,class)
```
This function creates the *mailboxName* Mailbox.

If the *class* is not a null-value, it specified the Mailbox Class for the newly created Mailbox.

This function returns a null-value if the Mailbox is successfully created, otherwise it returns a string with an error code.

```
RenameMailbox(oldMailboxName,newMailboxName,renameSub)
```
This function renames the *oldMailboxName* Mailbox into *newMailboxName*. Both parameters must have string values.

If the *renameSub* value is not a null-value, all *oldMailboxName* sub-Mailboxes are renamed, too.

This function returns a null-value if the Mailbox is successfully renamed, otherwise it returns a string with an error code.

```
DeleteMailbox(mailboxName,deleteSub)
```
This function deletes the *mailboxName* Mailbox.

If the *deleteSub* value is not a null-value, all *mailboxName* sub-Mailboxes are deleted, too.

This function returns a null-value if the Mailbox is successfully deleted, otherwise it returns a string with an error code.

```
GetMailboxACLs(mailboxName)
```
This function reads the Access Control List for the *mailboxName* Mailbox.

If the function successfully retrieves the ACL data, it returns a dictionary containing ACL identifiers as keys, and access rights as value strings.

Otherwise the function returns a string with an error code.

```
SetMailboxACLs(mailboxName,newACLs)
```
This function updates the Access Control List for the *mailboxName* Mailbox.

The *newACL* value should be a dictionary, containing ACL identifiers as keys, and access rights as value strings.

To remove an identifier from the ACL, specify an empty array as its value.

This function returns a null-value if the Mailbox ACL is successfully modified, otherwise it returns a string with an error code.

**Note:** To access Mailboxes in other Accounts, specify a Mailbox name as a `"~accountName[@domainName]/mailboxName"` string.

`GetMailboxAliases(accountName)`
> This function reads Mailbox Aliases created in the *accountName* Account or to the current Account, if the *accountName* value is a null-value.
>
> If the function successfully retrieves the Mailbox Aliases data, it returns a dictionary. Each dictionary key is the Mailbox Alias name, and its value is a string with the Mailbox name this Mailbox Alias points to.
>
> Otherwise the function returns a string with an error code.

`SetMailboxAliases(newAliases,accountName)`
> This function sets the Mailbox Aliases for the *accountName* Account or for the current Account, if the *accountName* value is a null-value. The old Mailbox Aliases are removed.
>
> The *newAliases* value should be a dictionary. Each dictionary key is the Mailbox Alias name, and its value is a string with the Mailbox name this Mailbox Alias points to.
>
> This function returns a null-value if the Mailbox Aliases are successfully modified, otherwise it returns a string with an error code.

`GetMailboxSubscription(accountName)`
> This function reads Mailbox Subscription created in the *accountName* Account or to the current Account, if the *accountName* value is a null-value.
>
> If the function successfully retrieves the Mailbox Subscription data, it returns an array. Each array element is a string containing a Mailbox name.
>
> Otherwise the function returns a string with an error code.

`SetMailboxSubscription(newSubscription,accountName)`
> This function sets the Mailbox Subscription for the *accountName* Account or to the current Account, if the *accountName* value is a null-value. The old Mailbox Subscription elements are removed.
>
> The *newSubscription* value should be an array. Each array element is a string containing a Mailbox name.
>
> This function returns a null-value if the Mailbox Subscription is successfully modified, otherwise it returns a string with an error code.

# Mailbox Handles

Mailbox handles are internal objects representing Mailboxes. Mailbox Messages can be addressed either using their UID (unique IDs) numeric values, or their current index values.

Some operations use a "message set" as a parameter. A message set specifies a set of numbers, which is interpreted either as a set of message UIDs or a set of message indeces.

If a "message set" parameter value is a number, the resulting set includes this number only.

If a "message set" parameter value is an array, containing numeric elements. All these numbers are included into the resulting set. The array can also contain "ranges", where a "range" is an array containing two numbers ("range ends"). The resulting set includes the "range end" numbers, and all numbers between them.

Examples:

> if a "message set" value is the `123` number, the resulting set includes the number 123 only
>
> if a "message set" value is the `(123, 125, 130)` array, the resulting set includes the numbers 123,125,130
>
> if a "message set" value is the `(123, (125,128), 130)` array, the resulting set includes the numbers 123,125,126,127,128,130

`OpenMailbox(mailboxName)`
> This function opens a Mailbox. The *mailboxName* value should be a string. It specifies the Mailbox name.
>
> If the name does not start with the ~ symbol, the Mailbox is opened in the current Account, if any.
>
> The current Account (if any) must have the Read/Select access right for the specified Mailbox.
>
> The function returns a Mailbox handle if the Mailbox has been opened successfully, otherwise it returns an error code

string.

`OpenMailboxView(`*`params`*`)`
> This function opens a Mailbox and creates a Mailbox handle.
>
> The *params* value should be a dictionary containing a `mailbox` and `sortField` string elements, and optional `mailboxClass`, `sortOrder`, `filter`, `filterField` string elements, and `UIDValidity`, `UIDMin` numeric elements.
>
> See the [XIMSS](#) section for more details on these parameter values.
>
> The function returns a Mailbox handle if the Mailbox has been opened successfully, otherwise it returns an error code string.

`MailboxUIDs(`*`boxRef`*`,`*`flags`*`)`
> This function returns an array of numbers - Mailbox message UIDs.
>
> The *boxRef* value should be a Mailbox handle.
>
> If the *flags* value is a string, it should contain a comma-separated list of message flag [Names](#) and/or [Negative Names](#). Only UIDs of messages that have flags specified with the flag Names and do not have flags specified with the Negative Names are included into the resulting array.
>
> The following example retrieves UIDs of all messages that have the Seen flag and do not have the Deleted flag:

```
myMailbox = OpenMailbox("INBOX");
seen = MailboxUIDs(myMailbox,"Seen,Undeleted");
```

`SubscribeEvents(`*`object`*`,`*`refData`*`)`
> This function enables or disables object *object* event generation.
>
> If the *refData* value is a null-value, the *object* stops to generate events.
>
> Otherwise, the *refData* value should be a ["basic"](#) object. When the *object* generates events and send them to this Task, the event *parameter* element contains the *refData* value.
>
> If the *object* value is a Mailbox handle, it should be created with the `OpenMailboxView` function. A Mailbox notification Event is sent to this Task when the Mailbox messages are removed, added, or modified. The "mailbox view" is not actually modified (i.e. the `MailboxUIDs` function returns the same set of message UIDs) until the Sync operation is applied to that Mailbox handle.

`IsMailboxNotifyEvent(`*`data`*`)`
> This function returns a true-value if the `data` value is a Mailbox notification Event, otherwise it returns a null-value.

`Sync(`*`boxRef`*`)`
> This function checks for Mailbox modifications.
>
> The *boxRef* value should be a Mailbox Handle.
>
> This function takes the first pending Mailbox modification and "commits" it by modifying the message UIDs (adding added messages, removing removed messages). It returns a dictionary with the following elements:

> mode
>> a `"removed"`, `"added"`, `"updated"`, or `"attrsUpdated"` string specifying this Mailbox modification type.

> UID
>> a number - the UID of the removed, added, or modified message

> index
>> for an updated message - its position in the message UIDs array.
>> for a removed message - its position in the message UIDs array before it was removed from that array.
>> for an added message - its position in the message UIDs array after it was added to that array.

> If there is no pending modification in the Mailbox, this function returns a null-value.

`MailboxInternalTimeByUID(`*`boxRef`*`,`*`uid`*`)`
> This function returns a timestamp object containing the message Internal Date.
>
> The *boxRef* value should be a Mailbox handle, the *uid* value should be a number - the message UID.
>
> If a message with the specified UID does not exist in the Mailbox, the function returns a null-value.

`MailboxFlagsByUID(`*`boxRef`*`,`*`uid`*`)`
> This function returns a string containing a comma-separated list of Mailbox message flags Names.
>
> The *boxRef* value should be a Mailbox handle, the *uid* value should be a number - the message UID.
>
> If a message with the specified UID does not exist in the Mailbox, the function returns a null-value.

`MailboxOrigUIDByUID(`*`boxRef`*`,`*`uid`*`)`

    This function returns a number containing the message "original" (permanent) UID.

    The *boxRef* value should be a Mailbox handle, the *uid* value should be a number - the message UID.

    If a message with the specified UID does not exist in the Mailbox, the function returns a null-value.

`MailboxUIDByOrigUID(`*`boxRef`*`,`*`origUID`*`)`

    This function returns a number containing the real UID of the message with the specified "original" (permanent) UID.

    The *boxRef* value should be a Mailbox handle, the *origUID* value should be a number - the message original UID.

    If a message with the specified original UID does not exist in the Mailbox, the function returns a null-value.

`MailboxSetFlags(`*`boxRef`*`,`*`idData`*`,`*`params`*`)`

    This function modifies Mailbox Message flags.

    The *boxRef* value should be a Mailbox handle, and the *idData* value is a "message set" (see above).

    If the *params* value is a string, it should be "flag values": a comma-separated list of message flag [Names](#) and/or [Negative Names](#).

    If the *params* value is a dictionary, it should include the following elements:

    flags
        the "flag values" string

    useIndex
        if this optional element is present, the "message set" specified with the *idData* value is interpreted as a set of message index values in the *boxRef* "mailbox view", if this element is absent, it is interpreted as the UID value set.
        This element can be present only if the *boxRef* handle was open using the OpenMailboxView function. Otherwise this function call results in a program exception.

    The function sets the flags specified with their Names and resets the flags specified with their Negative Names.

    The function returns a null-value if the current Account (if any) has sufficient Mailbox [Access Rights](#) to modify the flags, and flags have been successfully modified, otherwise the function returns an error code string.

`MailboxExpunge(`*`boxRef`*`)`

    This function removes all Mailbox messages marked as "purgable" or "deleted".

    The *boxRef* value should be a Mailbox handle.

    This function returns a null-value if the operation has succeeded, otherwise it returns a string with an error code.

`MailboxAudioByUID(`*`boxRef`*`,`*`uid`*`)`

    This function returns a datablock containing an audio section of the message.

    The *boxRef* value should be a Mailbox handle, the *uid* value should be a number - the message UID.

    If a message with the specified UID does not exist in the Mailbox, or the message does not contain an `audio` part, the function returns a null-value.

`MailboxRedirectByUID(`*`boxRef`*`,`*`uid`*`,`*`addresses`*`)`
`MailboxForwardByUID(`*`boxRef`*`,`*`uid`*`,`*`addresses`*`)`

    These function redirect or forward the specified message to the specified E-mail addresses.

    The *boxRef* value should be a Mailbox handle, the *uid* value should be a number - the message UID, the *addresses* should be a string containing one E-mail address or several E-mail addresses separated with the comma (`,`) symbols.

    These functions return a null-value if the operation has succeeded, otherwise they return a string with an error code.

`MailboxAppend(`*`boxRef`*`,`*`headers`*`,`*`content`*`)`

    These function composes an E-mail message and appends it to the Mailbox.

    The *boxRef* value should be a Mailbox handle.

    The *headers* and *content* values are processed in the same way they are processed with the [`SubmitEMail`](#) function.

    The *headers* dictionary may contain the following additional elements:

    flags
        a string that specifies the message flags the newly created message will have in the Mailbox. Several flags can be specified, separated with the comma symbol. See the [Mailbox](#) section for more details.

    internalDate
        a timestamp value with the "internal timestamp" for the newly created message. If absent, the current time is used.

    replacesUID, replaceMode

an UID value for the previous version of this message. The meaning is the same as the meaning of these attributes in the [XIMSS messageAppend](#) operation.

report
    if this element is specified, it should have the `"uid"` value.

If the *content* value is a [vCard](#) or [vCardGroup](#) XML object, a Contacts item is composed (regular *headers* elements are ignored), and this item is added to the Mailbox.

If the operation has failed, this function returns a string with an error code.

If the operation has succeeded, this function returns a null-value, or, if the `report` element was specified in the *headers* dictionary, the function returns a number - the UID value of the newly created Mailbox message.

MailboxCopy(*boxRef*,*idData*,*parameters*)
    These function copies or moves messages from one Mailbox to some other Mailbox.

    The *boxRef* value should be a Mailbox handle.

    The *uid* value should be either a number or an array of numbers or "ranges", where a "range" is an array with 2 numbers as its elements. The "range" includes its "start" and "end" numbers, and all the numbers in between.

    All these numbers specify the set of UIDs or Mailbox indeces of the message(s) to be copied.

    If the *uidData* value is the `123` number, the message with UID 123 is copied. If the *uidData* value is the `(123, 125, 130)` array, the messages with UIDs 123,125,130 are copied. If the *uidData* value is the `(123, (125,128), 130)` array, the messages with UIDs 123,125,126,127,128,130 are copied.

    If the *parameters* value is a string, it specifies the target Mailbox name.

    Otherwise, the *parameters* value should be a dictionary with the following elements:

    targetMailbox
        the target Mailbox name string.

    doMove
        if this element is present, the original messages are removed after they have been successfully copied.

    mailboxClass
        if this element value is a string, and the targetMailbox Mailbox does not exist, that Mailbox is created.
        If the element value is a non-empty string, that value is assigned as the [Mailbox Class](#) to the newly created Mailbox.

    useIndex
        if this element is present, the set specified with the *idData* value is interpreted as a set of message index values in the *boxRef* "mailbox view". This element can be present only if the *boxRef* handle was open using the OpenMailboxView function. Otherwise this function call results in a program exception.

    report
        if this element is present, the function returns an array of numbers - the UIDs of copied messages created in the target Mailbox.

This function returns a null-value or an array if the operation has succeeded, otherwise it returns a string with an error code.

GetMessageAttr(*boxRef*,*uid*)
    This function retrieves Message attributes.

    The *boxRef* value should be a Mailbox handle, the *uid* value should be a number - the message UID.

    If the function succeeds, it returns a dictionary with the message attribute values, otherwise it returns a string with an error code.

UpdateMessageAttr(*boxRef*,*uid*,*newData*)
    This function modifies Message attributes.

    The *boxRef* value should be a Mailbox handle, the *uid* value should be a number - the message UID.

    The *newData* value should be a dictionary with new attribute values. To remove an attribute, set its value to the `"default"` string.

    If the function succeeds, it returns a null-value, otherwise it returns a string with an error code.

# Message Handles

Message Handles are internal objects representing individual messages stored in [Mailboxes](#).

`OpenMessage(`*`boxRef`*`,`*`uid`*`)`
> This function opens a Message Handle.
> The *boxRef* value should be a Mailbox handle, the *uid* value should be a number - the message UID.
> The function returns a Message Handle if the message has been opened successfully, otherwise it returns a null-value.

`MessagePart(`*`msgRef`*`,`*`type`*`)`
> The *msgRef* value should be a Message Handle.
> If the `type` value is a null-value, this function returns a dictionary containing the message MIME structure.
> Otherwise, the `type` value should be a string. It specifies the message subpart to search for:
>
> `audio`
> > a subpart with `audio/*` Content-Type.
>
> `plain`
> > a subpart with `text/plain` Content-Type.
>
> `text`
> > a subpart with `text/*` Content-Type. If several alternative subtypes are available, the `text/plain` part has the lowest priority, followed by `text/html` part.
>
> If a subpart is found, this function returns a dictionary with this subpart MIME structure.
> If no subpart is not found, the function returns a null-value.
> If the message cannot be read or parsed, the function returns an error code string.
> The dictionary elements are:
>
> `estimatedSize`
> > estimated size of the message (or message part) decoded body.
>
> `filename`
> > optional - the decoded message part file name.
>
> `Content-Type`
> > the message (or message part) content type string (such as `"text"`, `"image"`, etc.)
>
> `Content-Subtype`
> > the message (or message part) content subtype string (such as `"plain"`, `"jpeg"`, etc.)
>
> `DispositionParams`
> > a dictionary with `Content-Disposition` field parameters
>
> `ContentTypeParams`
> > a dictionary with `Content-Type` field parameters
>
> `MIMEPartID`
> > This element is an empty string for the message itself. For message subparts, this element is a string with the message part ID.
>
> `MIMEParts`
> > an array of MIME subparts. Each array element is a dictionary with subpart MIME structure.
>
> *`contentFieldName`*
> > a string with `Content-`*`contentFieldName`* field value

`MessageHeader(`*`msgRef`*`,`*`partID`*`,`*`fieldName`*`)`
> This function returns an array of the specified message header fields, or a dictionary with all message header fields. If the message cannot be read or parsed, the function returns an error code string.
> The *msgRef* value should be a Message Handle.

If the *partID* is a null-value, then the message header fields are returned.

If the *partID* is a string, it should be a message part ID, and that message part should be of the `text/rfc822headers` type, or it should be the (only) subpart of the `message/rfc822` part. The header fields of the specified message part are returned.

If the *fieldName* value is a string, it specifies the message header field name, and the function value is an array of the specified field values.

Otherwise, the *fieldName* value should be a null-value. In this case the function value is a dictionary: the dictionary keys are the names of all message header fields, and their values are arrays with the corresponding field values.

If the field name specifies an Email-type field (`"From"`, `"To"`, `"Sender"`, etc.) then the field value is a dictionary. The dictionary element with an empty (`""`) key is the parsed E-mail address. An optional `realName` element contains the address "comment" string.

If the field name is a `"E-emailField"` string, where *emailField* is an Email-type field name, then the field value is a string with the parsed E-mail address.

If the field name is a date-type field (`"Date"`,`"Resent-Date"`), then the field value is a timestamp.

In all other cases, the field value is a string containing the field data, MIME-decoded and converted into the UTF-8 character set.

Example: the `msgRef` Message Handle references a message with the following header:

```
From: "Sender Name" <fromName@domain>
Subject: I'll be there!
To: "Recipient Name" <toName@domain>, <toName1@domain1>,
To: <toName2@domain2>
Cc: "Cc Name" <toName3@domain3>
MIME-Version: 1.0
X-Mailer: SuperClient v7.77
Date: Fri, 24 Oct 2008 02:51:24 -0800
Message-ID: <ximss-38150012@this.server.dom>
Content-Type: text/plain; charset="utf-8"
```

Then the `MessageHeader(msgRef,null,"To")` returns
`({""="toName@domain",realName="Sender Name"},{""="toName1@domain1"},{""="toName2@domain2"})`
the `MessageHeader(msgRef,null,"E-Cc")` returns
`("toName3@domain3")`
and the `MessageHeader(msgRef,null,null)` returns

```
{
 Cc = ({""="toName3@domain3",realName="Cc Name"});
 Date = (#T24-10-2008_10:51:24);
 From = ({""="fromName@domain",realName="Sender Name"});
 Message-ID = ("<ximss-38150012@this.server.dom>");
 Subject = ("I'll be there!");
 To=({""="toName@domain",realName="Recipient Name"},{""="toName1@domain1"},{""="toName2@domain2"});
 X-Mailer = ("SuperClient v7.77");
}
```

`MessageBody(msgRef,partID)`

This function returns the message or message part body.

The *msgRef* value should be a Message Handle.

If the *partID* is a null-value, then the entire message body is returned.

If the *partID* is a string, it should be a message part ID. The message part body is returned.

If the message cannot be read, or if the specified message part is not found, this function returns an error code string.

If the message or message part Content-Type is `text/*`, the function reencodes the result using the UTF-8 character set.

If the message or message part Content-Type is `text/xml`, the function returns a parsed XML object.

For all other Content-Types, the function returns a datablock with the message body data.

# Calendars

Calendar handles are internal objects representing Calendars. When a Calendar handle is created, it can be used to retrieve its events within the specified time intervals, to publish new events, to accept, decline, and cancel events, etc.

`OpenCalendar(mailboxName)`

This function opens the specified Mailbox as a Calendar. The *mailboxName* value should be a string. It specifies the Mailbox name.

If the name does not start with the ~ symbol, the Mailbox is opened in the current Account, if any.

The current Account (if any) must have the [Read/Select](#) access right for the specified Mailbox.

The function returns a Calendar handle if the Mailbox has been opened successfully and the Calendar is built using its content, otherwise the function returns an error code string.

`Find(calendarRef,params)`

the *calendarRef* value should be a Calendar handle, and the *params* value should be a dictionary.

This function retrieves all calendar events falling into the specified time interval.

The *params* dictionary should contain the required `timeFrom` and `timeTill` elements with the timestamp values, and optional `limit` and `skip` number elements, and an optional `byAlarm` element. These elements have the same meanings as the attributes of the <u>findEvents</u> XIMSS operation.

If the function fails, it returns an error code string. Otherwise, it returns an array of dictionaries for each 24-hour day (in the selected time zone) included into the specified time interval.

Each dictionary has the following elements:

`timeFrom`, `timeTill`, `skip`, `items`
> these elements have the same meanings as the attributes of the <u>events</u> XIMSS data message

`events`
> An array of found Events, represented as dictionaries with the following elements:

> `UID`, `timeFrom`, `dateFrom`, `duration`, `alarmTime`
>> these elements have the same meanings as the attributes of the `event` sub-elements of the <u>events</u> XIMSS data message

> `data`
>> the [XML presentation](#) of the found Event

> `parent`
>> this element is present if the found Event is an exception of some other Event. The element value is the [XML presentation](#) of that parent Event.
>> The value of the `data` element is included into the `parent` element value as a sub-element of its `exceptions` sub-element.

`ProcessCalendar(`*calendarRef*`,`*params*`)`
> the *calendarRef* value should be a Calendar handle, and the *params* value should be a dictionary.
> The function applies an operation specified with the *params* to the specified Calendar.
> If the operation succeeds, this function returns a null-value. Otherwise it returns an error code string.
> The operation performed is specified with the `opCode` string element of the *params* dictionary:

`"PUBLISH"`
> The function publishes an Event or a ToDo element in the Calendar. Other *params* elements:

> `data`
>> an <u>iCalendar</u> element to place into the Calendar.

> `attachments`
>> (optional) array containing the item attachments, each array element specified as an [EMail part content](#).
>> If this element is not specified, and a calendaring item with the same UID already exists in the Calendar, then all attachments are copied from the existing item. To remove all attachments, specify an empty array.

> `sendRequests`
>> this optional string element has the same meaning as the attribute of the XIMSS [calendarPublish](#) operation.

`"CANCEL"`
> The function removes an Event or a ToDo element from the Calendar. Other *params* elements:

> `data`
>> an <u>iCalendar</u> element to remove from the Calendar.

> `UID`
>> this optional numeric element has the same meaning as the attribute of the XIMSS [calendarCancel](#) operation.

> `itemUID`, `sendRequests`
>> these optional string elements have the same meanings as the attributes of the XIMSS [calendarCancel](#) operation.

recurrenceId
>this optional timestamp element has the same meaning as the attribute of the XIMSS [calendarCancel](#) operation.

requestComment
>this optional string element has the same meaning as the body element of the XIMSS [calendarCancel](#) operation.

"DECLINE"
>The function rejects a calendaring item and sends a negative reply the item organizer. Other *params* elements:

data
>an `iCalendar` element to decline.

sendReply
>this optional string element has the same meaning as the attribute of the XIMSS [calendarDecline](#) operation.

replyComment
>this optional string element has the same meaning as the body element of the XIMSS [calendarDecline](#) operation.

>The *calendarRef* value can be a null-value. In this case, only a negative reply is generated.

"ACCEPT"
>The function places a calendaring item into a Calendar and sends a positive reply the item organizer. The existing items(s) with the same UID are replaced. Other *params* elements:

data
>an `iCalendar` element to accept.

attachments
>this optional element has the same meaning here as for the PUBLISH operation code.

PARTSTAT, sendReply
>these string elements have the same meaning as the attributes of the XIMSS [calendarAccept](#) operation.

replyComment
>this optional string element has the same meaning as the body element of the XIMSS [calendarAccept](#) operation.

"UPDATE"
>The function updates a calendaring item in a Calendar using a reply-type `iCalendar` object. This item specifies if a particular attendee has accepted or rejected the invitation. Other *params* elements:

data
>an `iCalendar` "reply" element to use for updating.

# File Storage

The following functions manage files and file directories in the current Account [File Storage](#).
To access other Accounts File Storage, specify a file or folder name as a `~account[@domainName]/fileName` string.

ReadStorageFile(*fileDescr*)
>This function reads a File Storage file.
>If the *fileDescr* value is a string, it specifies the name of the [File Storage](#) file to read. The entire file is read - and only files not larger than 1MB in size can be read this way.

If the *fileName* value is a dictionary, the following dictionary elements are used:

fileName
> a string - the name of the file to read

position
> an optional element. Its value should be a non-negative number, it specifies the file position (in bytes) from which reading should start.

limit
> an optional element. Its value should be a non-negative number, it specifies the maximum data length to read (in bytes). If the file is shorter than starting position plus the read limit, a shorter datablock is read. This value should not exceed 1MB.

This function returns a datablock value with the file content, or a null-value if the specified file could not be read.

WriteStorageFile(*fileDescr*,*data*)
> This function stores the *data* value (which should be either a datablock or a string) into the specified [File Storage](#) file.
> If the *fileDescr* value is a string, it specifies the name of the file to write to. The entire file is rewritten.
> If the *fileName* value is a dictionary, the following dictionary elements are used:

fileName
> a string - the name of the file to write

position
> an optional element. If present, the file is not completely rewritten.
> If this element value is a non-negative number, it specifies the file position (in bytes) from which writing should start.
>
> If this element value is the `"end"` or `"append"` string, the writing operation starts from the current file end.
> If this element value is the `"new"` string, the writing operation checks that the file does not already exist.

This function returns a null-value if the file is successfully written, otherwise it returns a string with an error code.

AppendStorageFile(*fileName*,*data*)
> This function appends the *data* datablock or string to the end of the *fileName* [File Storage](#) file.
> If *fileName* is not a string or *data* is not a datablock nor it is a string, this function call results in a program exception.
> This function returns a null-value if the file is successfully written, otherwise it returns a string with an error code.

DeleteStorageFile(*fileName*)
> This function deletes the *fileName* [File Storage](#) file.
> This function returns a null-value if the file is successfully deleted, otherwise it returns a string with an error code.

RenameStorageFile(*oldFileName*,*newFileName*)
> This function renames the *oldFileName* [File Storage](#) file into *newFileName*. Both parameters must have string values.
> This function returns a null-value if the file is successfully renamed, otherwise it returns a string with an error code.

CreateStorageDirectory(*directoryName*)
> This function creates the *directoryName* [File Storage](#) directory.
> This function returns a null-value if the directory is successfully created, otherwise it returns a string with an error code.

RenameStorageDirectory(*oldDirectoryName*,*newDirectoryName*)
> This function renames the *oldDirectoryName* [File Storage](#) directory into *newDirectoryName*. Both parameters must have string values.
> This function returns a null-value if the directory is successfully renamed, otherwise it returns a string with an error code.

DeleteStorageDirectory(*directoryName*)
> This function deletes the *directoryName* [File Storage](#) directory. The directory should be empty.
> This function returns a null-value if the directory is successfully deleted, otherwise it returns a string with an error code.

ListStorageFiles(*folderName*)
> This function returns information about all files in the specified [File Storage](#) subdirectory.

If *folderName* is not a string, information about the top-level directory in the current Account [File Storage](#) is returned.

This function returns a null-value if an error occurred. Otherwise, the function returns a dictionary. Each dictionary key is a file or a subdirectory name, and the dictionary value is a dictionary with the following elements:

STFileSize
> a numeric value with the file size in bytes. This element is present for files and absent for subdirectories.

STCreated
> (optional) a timestamp value with the file creation date.

STModified
> a timestamp value with the file modification date.

MetaModified
> (optional) a timestamp value with the file or subdirectory attribute set modification date.

ReadStorageFileAttr(*fileName*,*attributes*)
> This function reads attributes of the *fileName* [File Storage](#) file or file directory.
> If *fileName* is not a string or *attributes* is neither an array nor a null-value, this function call results in a program exception.
> The *attributes* value should be either a null-value (then all file attributes are retrieved), or an array of strings - then only the attributes with names included into the array are retrieved.
> If attributes are successfully retrieved, this function returns an array of file attributes (each attribute is an XML object).
> Otherwise the function returns a string with an error code.

WriteStorageFileAttr(*fileName*,*attributes*)
> This function modifies attributes of the *fileName* [File Storage](#) file or file directory.
> If *fileName* is not a string or *attributes* is not an array, this function call results in a program exception.
> The *attributes* value should be an array of XML objects. See the [XIMSS](#) section for more details.
> This function returns a null-value if the attributes have been successfully updated, otherwise it returns a string with an error code.

LockStorageFile(*fileName*,*params*)
> This function manages "locks" of the *fileName* [File Storage](#) file or file directory.
> If *fileName* is not a string or *params* is not a dictionary, this function call results in a program exception.
> The *params* value should be a dictionary with lock request parameters. See the [XIMSS](#) section for more details.
> This function returns a dictionary if the operation has been completed successfully, otherwise it returns a string with an error code.

# Roster

The following built-in functions implement an interface to the Account Roster.

ReadRoster()
ReadRoster(*accountName*)
> This function retrieves Roster items.
> If the the *accountName* parameter is not specified, or if its value is a null-value, the current Account Roster is read, otherwise the *accountName* value should be a string specifying the name of the Account to read Roster items from.
> This function returns a dictionary with Roster items, where dictionary keys are contact E-mail addresses, and dictionary values are dictionaries with the following elements:

Inp
> the "incoming" subscription mode: it controls if the contact can watch the user presence. Possible values: null-value, true-value, `"Pending"`, `"Blocked"`.

Out
> the "outgoing" subscription mode: it controls if the user can watch the contact presence. Possible values: null-value, true-value, `"Pending"`.

RealName
> the contact real name string (optional).

If the function fails to retrieve Roster items, it returns an error code string.

```
SetRoster(params)
SetRoster(params,accountName)
```
This function updates a Roster item.

If the the *accountName* parameter is not specified, or it value is a null-value, the current Account Roster is updated, otherwise the *accountName* value should be a string specifying the name of the Account to update.

The *params* should be a dictionary with the following elements:

peer
> The contact E-mail address string (`"accountName@domainName"`).

what
> The operation type string:

- `"update"`: update the contact information.
- `"remove"`: remove the contact from the Roster.
- `"subscribed"`: confirm the contact request to monitor the user's presence information.
- `"unsubscribed"`: reject the contact request to monitor the user's presence information, or revoke the already granted right to monitor.
- `"subscribe"`: send a request to monitor the contact's presence information.
- `"subBoth"`: confirm the contact request to monitor the user's presence information, and send a request to monitor the contact's presence information.
- `"unsubscribe"`: stop monitoring the contact's presence information.

data
> optional - a dictionary containing new contact info.

# Datasets

The following built-in functions implement an interface to the Account Datasets. To access Datasets in other Accounts, the dataset name should be specified as a `"~accountName[@domainName]/datasetName"` string.

```
DatasetList(datasetName,filterField,filterValue)
```
This function retrieves data entries from an Account dataset.

The *datasetName* value should be a string with the dataset name.

The *filterField*, *filterValue* values should be null-values or strings. If the values are strings, they specify an entry attribute name and value. Only the entries that have the specified attribute with the specified value are included into the resulting value.

This function returns a dictionary with dataset entries. The dictionary keys are entry names, the dictionary elements are data entries. Each data entry is a dictionary containing the data entry attributes.

If the dataset entries could not be retrieved, the function returns an error code string.

```
DatasetCreate(datasetName)
```
This function creates an Account dataset.

The *datasetName* value should be a string; it specifies the dataset name.

If the dataset is created successfully, the function returns a null-value. Otherwise the function returns an error code string.

```
DatasetRemove(datasetName,ifExists)
```
This function removes an Account dataset.

The *datasetName* value should be a string; it specifies the dataset name.

If the *ifExists* value is not a null-value, and the dataset to be removed already does not exist, the function does not return an error code.

If the dataset is removed successfully, the function returns a null-value. Otherwise the function returns an error code string.

```
DatasetSet(datasetName,entryName,entryData,ifExists)
```
This function modifies data entries in an Account dataset.

The *datasetName* value should be a string; it specifies the dataset name.

The *entryName* value should be a string; it specifies the entry name.

The *entryData* value should be a dictionary; it specifies the entry data (attributes).

If the *ifExists* value is a null-value, then the dataset is created if it does not exist, and the entry with the specified name is created if it does not exist.

If the dataset is updated successfully, the function returns a null-value. Otherwise the function returns an error code string.

`DatasetDelete(`*`datasetName`*`,`*`entryName`*`,`*`ifExists`*`)`

This function deletes a data entry from an Account dataset.

The *datasetName* value should be a string; it specifies the dataset name.

The *entryName* value should be a string; it specifies the entry name.

If the *ifExists* value is not a null-value, and the entry to be deleted already does not exist, the function does not return an error code.

If the dataset is removed successfully, the function returns a null-value. Otherwise the function returns an error code string.

## Directory

The following built-in functions implement an interface to the Directory Manager.

`DirectorySearch(`*`baseDN`*`,`*`filter`*`,`*`parameters`*`)`

This function performs a directory search, returning a dictionary with found records.

The *baseDN* value should be a string with the search base DN. If this value is "$", the Directory Integration settings are used to compose the base DN for the current Domain.

The *filter* value should be a null-value, or a string with a search filter, in the RFC2254 format.

The *parameters* value should be a null-value or a dictionary containing search options:

limit
> If this option value is a positive number, it specifies the maximum number of records to return. Otherwise, the record limit is set to 100.

keys
> If this option value is `"DN"`, the resulting dictionary keys are full record DNs. Otherwise, the record RDNs are used as resulting dictionary keys.

scope
> If this option value is `"sub"`, a subtree search is performed. Otherwise, only the direct children of the base DN record are searched.

attributes
> If this option value is an array of strings, only the attributes listed in the array are included into resulting records. Otherwise, all record attributes are included into resulting records.

If the directory search operation fails (no base DN record, insufficient access rights, etc.), this function returns an error code string.

## Services

`GetLanguage()`

This function value is a string with the currently "selected language".

`SetLanguage(`*`lang`*`)`

This procedure sets the "selected language". The *lang* value should be a string with the language name, or a null-value to select the default language.

`GetTimeZoneName()`

This function value is a string with the currently selected time zone name. If no time zone is selected (the Server time offset is used), the function returns a null-value.

`SetTimeZone(`*`zoneName`*`)`

This procedure sets the current time zone. The *zoneName* value should be a string with a known time zone name. If an

unknown zone name is specified, or if the *zoneName* value is not a string, the *no zone* fictitious value is set, and the Server current time offset is used.

`ReadEnvirFile(`*fileName*`)`
> This function retrieves the specified file from the current "environment" - such as the [Real-Time Application](#) environment. This function value is a datablock with the file content or a null-value if the specified file does not exist.

`ReadSkinObject(`*tagName*`)`
`ReadSkinObject(`*tagName*`,`*index*`)`
`ReadSkinObject(`*tagName*`,`*index*`,`*skinName*`)`
> This function retrieves the specified object from the specified WebUser skin's [Text Dataset](#) (or the Basic skin, if *skinName* is not specified). Object is specified by its *tagName* and *index*, where *index* can be either of:

> `null`
>> the entire object is returned;

> *a string*
>> used as the key when the object is a dictionary;

> *a number*
>> used as the index when the object is an array or a dictionary;

> *an array*
>> elements are sequentially used as keys or indices through the object hierarchy.

> This function value is the object defined by the combination of *tagName* and *index* or a null-value if the specified object does not exist in the skin's Text Dataset.

`SysLog(`*arg*`)`
> This procedure places the textual representation of the *arg* value into the Server Log.

`SysProfile(`*arg*`)`
> If the *arg* value is a null-value, the procedure decreases an internal integer *profile level* counter by 1, otherwise it increases it by 1.
> The *profile level* counter is set to zero when the program starts.
> When the *profile level* counter is positive, profiling records are placed into the Server Log when every user-defined and some built-in functions and procedures are entered and exited.

`ExecuteCLI(`*arg*`)`
> This function executes the [Command Line Interface (CLI)](#) command.
> The *arg* value should be a string containing one CLI command.
> If the CLI command has been executed successfully, this function returns a null-value. Otherwise this function returns an error code string.
> If the CLI command has produced an output, it is placed into the [Task variable](#) `executeCLIResult` (it can be accessed as `Vars().executeCLIResult`.
> If the CLI command has failed, or it has not produced any output, this Task variable is assigned a null-value.

`SetApplicationStatus(`*arg*`)`
> This procedure copies its argument and stores is in the current Task descriptor.
> External modules and entities can retrieve this information from the Task descriptor.

`StoreCDR(`*arg*`)`
> This procedure sends the *arg* value (which should be a string) to the [External CDR Processor](#) program.
> The `"APP"` prefix is added to the *arg* value, and the application is supposed to provide its name and the version number as the first part of the *arg* value to allow the External CDR Processor program to differentiate records generated with different applications:

>> `APP `*arg*

`DoBalance(`*parameters*`,`*accountName*`)`
> This function performs a [Billing](#) operation.
> If the *accountName* is null, the operation is applied to the current Account. Otherwise, the *accountName* value must be a

string specifying the target Account name. In any case, the operation is subject to the Access Rights restrictions.

The *parameters* value must be a dictionary. It specifies the operation parameters, see the [Billing](#) section for the details. The operation name should be specified as the value of the dictionary key `"op"`.

The function returns an error code string if the operation has failed.

Otherwise the function returns a dictionary with the operation results (as specified in the [Billing](#) section).

Statistics(*elementName*,*opCode*,*setValue*)

This function reads and updates the Server [Statistics Element](#) value.

The *elementName* value should be a string specifying either a [Statistics Element](#) OID, or a [Custom Statistics Element](#) name.

The *setValue* value should be a numeric value.

The *opCode* value should be a null-value or a string. If the string value is `"inc"`, then the Element value is increased by the *setValue* value. If the string value is `"set"`, then the Element value is set to the *setValue* value. If the value is a null-value, then the Element value is not modified.

Only the Custom Statistics Elements can be modified.

The function returns either a number - the current Statistics Element value, or an error code string.

CallHelper(*name*,*parameters*)

This function sends a request to an [External Application Helper](#).

The *name* value should be a string specifying the Application Helper name.

The *parameters* value (application-specific) is passed to the Application Helper program.

The function returns the data object from the External Application Helper response (it may return a null-value).

BannerRead(*type*,*parameters*)

This function sends a request to the [External Banner System](#).

The *type* value should be a string specifying the banner type (application-specific, such as `"prontoEmailTop"`, `"myClientLeftBanner"`).

The *parameters* value is passed to the External Banner System.

The function returns the banner data object from the External Banner System response (it may return a null-value).

# Communications

HTTPCall(*URL*,*parameters*)

This function performs an HTTP transaction.

The current Account must have the `HTTP` [Service](#) enabled.

The *URL* value should be a string. It specifies the request URL.

The *parameters* value should be either a null-value or a dictionary. It specifies the request parameters and, optionally, the request body.

These values, along with the maximum time-out time of 30 seconds are passed to the [HTTP Client](#) module for processing.

This function returns either a result dictionary the HTTP module produced, or an error code string.

SubmitEMail(*headers*,*content*)

This function composes and sends an E-mail message.

The *headers* value should be a null-value or a dictionary. This dictionary specifies the header field values for the composed E-mail message. The following elements are processed (all of them are optional):

From

the element value should be an E-mail address string. It specifies the message `From:` address

To

the element value should be an E-mail address string or an array of E-mail address strings. It specifies the message `To:` address(es)

Cc

the element value should be an E-mail address string or an array of E-mail address strings. It specifies the message `Cc:` address(es)

Bcc

the element value should be an E-mail address string or an array of E-mail address strings. It specifies the message `Bcc:` address(es)

Subject

the element value should be a string. It specifies the message `Subject:` field.

Date

the element value should be a timestamp. It specifies the message `Date:` field. If this element is absent, the current time is used.

sourceType

the element value should be a string. It specifies the [message source](). If this element is absent, the `"CGPL"` string value is used.

sourceAddress

the element value should be a string. It specifies the address of the message source (network address, remote system name, etc.)

Message-ID

the element value should be a string. It specifies the message `Message-ID:` field. If this element is absent, an automatically generated string is used.

protocol

the element value should be a string. It specifies the name of the protocol used to submit this message.

Content-Class

the element value should be a string. It specifies the E-mail header `Content-Class:` field value.

X-Priority

the element value should be a string. It specifies the E-mail header `X-Priority:` field value.

X-Mailer

the element value should be a string. It specifies the message `X-Mailer:` field. If this element is absent, an automatically generated string is used.

The *content* value specifies the E-mail message body. If the value is a dictionary, then the dictionary `body` element is the actual content, and other dictionary elements specify various body parameters (content header fields). Otherwise the *content* itself is the actual content and the content body parameters set is an empty one.

The following content body parameters (dictionary elements) are processed (all of them are optional):

Content-Type

the element value should be a string. It specifies the content body Content-Type. If this element is not specified, the Content-Type is set to `"text"` if the actual content is a string, otherwise it the Content-Type is set to `"application"`

Content-Subtype

the element value should be a string. It specifies the content body Content-Type subtype. If this element is absent, and the Content-Type is set to `"text"`, the Content-Subtype is set to `"plain"`

filename

the element value should be a string. It specifies the content body file name.

Content-Disposition

the element value should be a string. It specifies the content body Content-Disposition. If this element is absent, and the `fileName` element is present, the Content-Disposition value is set to `"attachment"`

If the actual content is a string, it is stored "as is", using the `8bit` Content-Transfer-Encoding.
If the actual content is a datablock, it is stored using the `base64` Content-Transfer-Encoding.
If the actual content is an array, the content is a multi-part one. Only the Content-Subtype parameter element is used, if it is absent, the `"mixed"` value is used.
Each array element is stored in the same way as the *content* value itself.
If the actual content is not an array, a string, or a datablock, an empty content body is stored.
This function returns a null-value if an E-mail message has been composed and sent (submitted to the Queue). Otherwise, this function returns an error code string.

In the following example, a simple text message is sent.

```
    headers = NewDictionary();
    headers.From    = "from@sender.dom";
    headers.Subject = "Test Message";
    headers.To      = "To@recipient.dom";
    result  = SubmitEmail(headers,"Test Message Body\eEnd Of Message\e");
```

In the following example, a multipart/mixed message is sent. It contains an HTML text and a binary attachment.

```
    content = NewArray();

    textPart = NewDictionary();
    textPart.("Content-Type") = "text";
    textPart.("Content-Subtype") = "html";
    textPart.body = "<HTML><BODY>This is an <B>HTML</B> text</BODY></HTML>";
    content[0] = textPart;

    dataPart = NewDictionary();
    dataPart.("Content-Type")    = "image";
    dataPart.("Content-Subtype") = "gif";
    dataPart.fileName = "file.gif";
    dataPart.body     = ReadStorageFile(dataPart.fileName);
    content[1] = dataPart;

    headers = NewDictionary();
    headers.From    = "from@sender.dom";
    headers.Subject = "Test Attachment";
    headers.To      = "To@recipient.dom";
    headers.("Content-Class") = "message";

    result = SubmitEMail(headers,content);
```

It is possible to include parts of other E-mail messages into a newly composed one. If there is a content body parameter `MIMEPartID` with a string value, then there must be a content body parameter `source` with a Message Handle value.
If the `MIMEPartID` parameter value is `"message"`, the entire `source` Message is copied (as a `message/rfc822` MIME part body).
If the `MIMEPartID` parameter value is any other string, it specifies the MIME part of the `source` Message to be copied into the new message; when copying the part headers, only the MIME (`Content-xxxxxx`) fields are copied.

```
    content = NewArray();

    textPart = NewDictionary();
    textPart.("Content-Type")    = "text";
    textPart.("Content-Subtype") = "plain";
    textPart.body = "Please see the attached letter.\e";
    content[0] = textPart;

    attachPart = NewDictionary();
    attachPart.("source")    = MyOldMessage;
    attachPart.("MIMEPartID") = "message";
    content[1] = dataPart;

    headers = NewDictionary();
    headers.From    = "from@sender.dom";
    headers.Subject = "Test Forwarded message";
    headers.To      = "To@recipient.dom";
    headers.("Content-Class") = "message";

    result = SubmitEMail(headers,content);
```

SendEMail(*fromAddress*,*subject*,*to*,*headers*,*content*)
> This function composes and sends an E-mail message. A call to this function does the same as the SubmitEMail(*headers*,*content*) function call, where the *fromAddress*, *subject*, and *to* value (if they are not null-values) are used instead of the *headers* dictionary From,Subject, and To elements.

> In the following example, a simple text message is sent.

```
    result = SendEmail("from@sender.dom","Test Message","To@recipient.dom",
                       null,"Test Message\eEnd Of Message\e");
```

SendInstantMessage(*fromAddress*,*toAddress*,*content*)
> This function sends an Instant Message.
> The *fromAddress* value should be a string or a null-value. It specifies the message From (sender) address.
> If this value is a null-value, then the sender address is the current Account address.
> If this value is a string, but it does not contain a @ symbol, the sender address is the current Account address, and this value

is used as the sender "instance" name.

If this value contains a `@` symbol, it can also specify the sender "instance" name, as in `"user@domain/instance"` string.

The *toAddress* value should be a string. It specifies the message `To` (recipient) address.

It must contain the `@` symbol and it can specify the recipient "instance" name, as in `"user@domain/instance"` string.

The *content* value should be a string or a dictionary. If the value is a string, it specifies the message content. If the value is a dictionary, it can contain the following elements:

`""` (empty string)
> this optional element should be a string - the instant message body.

`type`
> this optional element should be a string - the instant message XMPP-style type - `"chat"`, `"groupchat"`, etc.

`id`
> this optional element should be an string - it specifies the IM request "id" attribute.

`suppl`
> this optional element should be an array of XML objects to be sent inside the instant message.

`IMSubject`
> this optional element should be a string - the instant message subject.

`IMState`
> if this optional element is present, the `""` (body), IMSubject and suppl elements should not be used. This element should be one of the strings `"gone"`, `"composing"`, `"paused"`, `"active"`, and informs the recipient about the sender intentions.

The function only initiates an Instant Message signaling operation, it does not wait for the message delivery operation to complete.

This function returns a null-value if an Instant Message has been composed and sent (submitted to the Signal component). Otherwise, this function returns an error code string.

`SendXMPPIQ(`*fromAddress*`,`*toAddress*`,`*content*`)`
> This function sends an XMPP-style IQ request or a presence request.
>
> The *fromAddress* and *toAddress* values have the same meaning as for the SendInstantMessage function.
>
> The content value should be a dictionary with the following elements:

`type`
> this element should be a string - `"get"`, `"set"`, `"result"`, etc. To send a presence request, this element should be `"cgp_presence"`.

`suppl`
> this element should be an array of XML objects to be sent inside the IQ request.

`id`
> this element should be an string - it specifies the IQ request "id" attribute. This element is not used for presence requests.

`opcode`
> this element should be an string, it is used for presence requests only. It specifies the presence request "type" attribute value, such as `"unavailable"`.

`status`
> this element should be an string, it is used for presence requests only. It specifies the presence state, such as `"online"`, `"busy"`, etc.

The function only initiates an signaling operation, it does not wait for the operation to complete.

This function returns a null-value if a request has been composed and sent (submitted to the Signal component). Otherwise, this function returns an error code string.

RADIUSCall(*address*,*parameters*)

      This function composes and sends a RADIUS request.

      The *address* value should be an ip-address. It specifies the address and the port of the RADIUS server to send the request to.

      The *parameters* value should be a dictionary with the following elements:

Type

      this element should be a string. It specifies the type of RADIUS request to send:

            `"authenticate"`, `"accountingStart"`, `"accountingUpdate"`, `"accountingStop"`.

Secret

      this element should be a string. It specifies the "shared secret" of the RADIUS server.

Username

      this element should be a string. It is used to compose the `userName` (1) request attribute.

Password

      this element should exist in the `authenticate` requests, and it should be a string. It contains the clear text password to use in the authentication operation.

*nn* (where nn is a decimal number)

      these elements specify additional request attributes (by their attribute numbers).

      These element values should be strings, or (for the integer-type parameters) - numbers, or (for Internet-type parameters) ip-addresses.

      If an element value is a datablock, then the content of the datablock is sent without any encoding.

      If an element value is an array, then the request attribute is added to the request zero or more times, once for each array value.

*-nnnnn* (where nnnnn is a decimal number)

      these elements specify additional vendor-specific attributes (where nnnnn is the VendorID). See the [RADIUS](#) section to see more on the vendor-specific attribute presentation.

Error

      if this optional element value is `"data"`, and a denied-response is received for an authentication request, the function returns a response attribute dictionary (see below) with an additional `Error` element.

      If this optional element value is not `"data"`, and a denied-response is received for an authentication request, the function returns an error string.

Retries

      this optional element should be a number or a numeric string in the [1..20] range. It overrides the default limit of request resends. Set this element to 1 if you want the RADIUS request to be sent only once.

Timeout

      this optional element should be a number or a numeric string in the [0..30] range. It overrides the default timeout value (in seconds). When a response is not received within the specified time, the request is resent or an error message is produced.

If the RADIUS operation succeeds, this function returns a dictionary. This dictionary contains attributes the RADIUS server sent in its response. If the RADIUS fails, this function returns a string with an error code.

**Note:** requests are sent using the UDP socket of the [RADIUS module](#), so this module should be enabled (it should be configured to use some non-zero port number).

In the following example, a RADIUS authentication request is sent. The request contains the nasIdentifier (32) text attribute.

```
callParam = newDictionary();
callParam.Type     = "authenticate";
callParam.Secret   = "sys2";
callParam.Username = "user4567";
callParam.Password = "drum$1245";
callParam.("32")   = "my NAS";
result = RADIUSCall(IPAddress("[10.0.1.77]:1812"),callParam);
```

In the following example, a RADIUS accounting request is sent. The request contains the nasIdentifier (32) text attribute and the acctSessionTime (46) numeric attribute.

```
callParam = newDictionary();
callParam.Type     = "accountingStart";
callParam.Secret   = "sys2";
callParam.Username = "user4567";
callParam.("32")   = "my NAS";
callParam.("46")   = SessionTimeInMinites*60;
result = RADIUSCall(IPAddress("[10.0.1.77]:1812"),callParam);
```

# Multitasking

Certain environments (such as Real-Time Application environments) provide multitasking functionality. In these environments, program invocations (*Tasks*) can locate each other and exchange data. This section defines the additional language features available in these multitasking environments.

Task handles are internal objects representing a Task. In a Cluster environment, a Task handler includes a reference to the cluster member running the Task.

## Spawning

A program (a running Task) can create a new Task by using the spawning expression. It is specified using the spawn keyword followed by a name of an entry code section. A new Task is created and it starts to run concurrently with the Task that used the spawning expression, executing the specified entry code section.
The entry code section name can be followed with an expression enclosed in parentheses. If specified, the copy of the expression value is assigned to the startParameter element of the Vars() dictionary (global variables) in the new Task.
The current Task handle is assigned to the parent element of the Vars() dictionary in the new Task.
The spawning expression value is a Task handle for the newly created Task, or null if the system failed to create a new Task.

In the following example, a program executing the Main entry code section creates a new task that starts to execute the DoBackup entry code section, which copies files "file1","file2",...."file100" into "backup1","backup2",..., files.

```
entry DoBackup is
  nameIndex = 1;
  while nameIndex <= 100 loop
    fileData = ReadStorageFile("file" + String(nameIndex));
    if fileData != null then
      resultCode = WriteStorageFile("backup" + String(nameIndex),fileData);
      if resultCode != null then
        Log("failed to backup file" + String(nameIndex) +
            ". Error Code=" + resultCode);
      end if;
    end if;
  end loop;
end entry;

entry Main is
  backuper = spawn DoBackup;
  if backuper == null then
    Log("Failed to start a Backup Task");
  end if;
end entry;
```

Tasks do not share any variables, even when a Task directly creates a new Task using the spawning expression.

ThisTask()
>    This function returns the Task handle of the current Task.
>    This function returns a null value if the current environment is not a Task.

IsTask(*arg*)
>    This function returns a true-value if the *arg* value is a Task Handle, otherwise the function returns a null-value.

## Events

The *Actor Model* is used: Tasks exchange data by sending each other *Events*.

```
SendEvent(taskRef,eventName)
SendEvent(taskRef,eventName,eventParam)
```
This function sends an Event to a Task. It returns a null value if an Event was sent successfully, or a string with an error code otherwise (for example, when the specified Task does not exist).

The *taskRef* value should be a Task handle.

The *eventName* value should be a string starting with a Latin letter.

The optional *eventParam* parameter value should be a null-value, or a Task handle, or a "basic" object.

This function only sends an Event to the specified (target) Task. It does not wait till that Task receives an event, nor does it wait for any response from the target Task.

```
ReadInput(secsToWait)
```
Events sent to a task are enqueued, and the task can read the first Event in queue using this function. The function value is a dictionary containing the event data.

The *secsToWait* value should be a non-negative number.

If the Task Event queue is empty and the Task does not receive a new Event within the specified number of seconds, the function returns a null-value.

If this function returns a dictionary value, the dictionary contains the following elements:

what
> the Event name string. If the Event was sent using the SendEvent operation, this string is the *eventName* parameter value used in the SendEvent call in the sender Task.

sender
> the Task handle of the sender Task (the Task that has sent this event). In a Cluster environment, this Task and the current Task may be running on different Cluster member computers.
> If the Event is sent by the platform itself, or if the sender was not a Task, the `sender` element does not exist.

parameter
> the event parameter. If the event was sent using the SendEvent operation in the sender Task, this element contains the *eventParam* parameter value of the SendEvent call.

**Note:** depending on the environment, the ReadInput function can return various other objects. For example, if the function is used in a Real-Time Application environment, it can return a string containing the first enqueued DTMF symbol.

**Note:** the ReadInput function may have "false wakeups", i.e. it can return the null-object even before the specified time period has elapsed.

# Meetings

The Meeting mechanism allows a Task to make itself known to other Tasks associated with the same Account.

Each Account can have several named Meeting Sets and an unnamed Default Meeting Set. Several named Meetings can be created in any Meeting Set. Each Meeting is a dictionary object that can contain zero or one Task handle.

The set name can be specified as a `"~accountName[@domainName]/setName"` string to access Meetings Sets in other Accounts. To access Meeting Sets in other Accounts, the current Account should have the `canImpersonate` Domain Access Right for the target Account Domain.

```
CreateMeeting(setName,key)
CreateMeeting(setName,key,parameter)
```
This function creates a Meeting object in some Meeting Set within the current Account.

The *setName* parameter specifies the Meeting set name. If its value is a null-value or an empty string, the Default Meeting Set is used.

The *key* parameter must be a string. It specifies a unique ID or name for the new Meeting. For example, an application implementing real-time conferencing can generate a random numeric string to be used as the conference password, and it can create a Meeting using that string. Other Tasks associated with the same Account can find that Meeting (and a Task handle for the Task associated with it) if they know the *key* and the *setName* parameter values used with the CreateMeeting operation.

If the *parameter* parameter is specified and its value is not a null-value, that value is stored with the Meeting. Note that the

value is stored using its textual representation, so only the standard objects can be used as the *parameter* values or the *parameter* value sub-elements. For example, you cannot store Mailbox or Task handles.

This function returns a null-value if a Meeting has been created. Otherwise, this function returns an error code string.

```
ActivateMeeting(setName,key)
ActivateMeeting(setName,key,parameter)
```
> This function adds the Task handle for the current Task to a Meeting in the current Account.
>
> The *setName* and *key* parameter values specify an already created Meeting.
>
> There should be no other Task handle stored in this Meeting.
>
> The current Task becomes the Meeting *Active Task*.
>
> The optional *parameter* parameter value is stored with the Meeting.
>
> This function returns a null-value if the Task handle has been successfully added. Otherwise, this function returns an error code string.

```
DeactivateMeeting(setName,key)
DeactivateMeeting(setName,key,parameter)
```
> This function removes Task handle for the current Task from a Meeting in the current Account.
>
> The *setName* and *key* parameter values specify an already created Meeting, and that Meeting should contain a Task handle for the current Task.
>
> The optional *parameter* parameter value is stored with the Meeting.
>
> When this operation completes successfully, the Meeting has no *Active Task*.
>
> This function returns a null-value if the Task handle has been successfully removed. Otherwise, this function returns an error code string.

```
RemoveMeeting(setName,key)
```
> This function removes a Meeting from a current Account Meeting Set.
>
> The *setName* and *key* parameter values specify the Meeting to remove.
>
> This function returns a null-value if the Meeting has been successfully removed or if the specified Meeting has not been found. Otherwise, this function returns an error code string.

```
FindMeeting(setName,key)
```
> This function retrieves Meeting information from a current Account Meeting Set.
>
> The *setName* and *key* parameter values specify the Meeting to look for.
>
> If the Account does not have the specified Meeting, the function returns null.
>
> Otherwise, the function returns the Meeting information dictionary containing the following elements:
>
> parameter
>> the value of the *parameter* parameter used with the CreateMeeting operation.
>
> id
>> a Task handle for the *Active Task*, if any.
>>
>> In a Cluster environment, the Active Task and the current Task may be running on different Cluster member computers.

```
ClearMeeting(setName,key)
```
> This function removes a Task handle from a Meeting (if any).
>
> The *setName* and *key* parameter values specify the Meeting to clear.
>
> This function can be used to clear a reference set by a Task that has died.
>
> This function returns a null-value if the Meeting has been successfully cleared. Otherwise, this function returns an error code string.

# Queues

The Queue mechanism allows a Task to make itself known to other Tasks associated with the same Account. When a Task registered in a Queue is found by some other Task, the found Task is removed from the Queue.

The queue name can be specified as a `"~accountName[@domainName]/queueName"` string to access Queues in other Accounts. To access Queues in other Accounts, the current Account should have the `canImpersonate` Domain Access Right for the target Account Domain.

```
Enqueue(queueName)
Enqueue(queueName,parameter)
Enqueue(queueName,parameter,pty)
```
This function registers the current Task with the Account associated with it.

An Account may have several Queues with Task registrations.

The *queueName* parameter specifies the Queue name. If this parameter value is a null-value or an empty string, the default Queue of the associated Account is used.

The optional *parameter* parameter value is stored with the Task registration. Note that the value is stored using its textual representation, so only the standard objects can be used as the *parameter* values or the *parameter* value sub-elements. For example, you cannot store Mailbox or Task handles.

The optional *pty* parameter value should be a string containing a decimal number with one digit, the dot (`.`) symbol and any number of digits. The Task is placed in the Queue before all other Tasks with a smaller *pty* parameter value, but after all tasks with the same or larger *pty* parameter value.

If the *pty* parameter is not specified, or if its value is a null-string, the default `"1.0"` value is assumed.

A Task may register itself several times in different Queues, but it can be registered only once with any given Queue. If the Enqueue function is used by a Task that has been already enqueued into the same Queue, the function does not create a second registration. Instead, the function updates the *parameter* value enqueued with the Task and may change the Task position in the Queue according to the new *pty* parameter value.

The function returns a null-value if registration has failed. If registration was successful, the function returns a dictionary containing the following elements:

length
>    a number - the total number of Tasks in the Queue

position
>    a number - the current position of the current Tasks in the Queue.
>    The position of the first Task in the Queue is `0`

```
CheckQueue(queueName)
```
This function checks the current Task position in an Account Queue.

The *queueName* parameter specifies the Queue name.

The function returns a null-value if it has failed to access the specified Queue. Otherwise, it returns the same dictionary as the Enqueue function.

**Note:** the `position` element exists in the returned dictionary only if the current Task is currently enqueued into the specified Queue. If current Task was not enqueued, or if it has been already removed from the Queue by some other task, this elements will be absent.

```
Dequeue(queueName)
```
This procedure removes the current Task from the Account Queue.

The *queueName* parameter specifies the Queue name.

```
ReadQueue(queueName)
```
This function retrieves the first Task from the Account Queue.

The *queueName* parameter specifies the Queue name.

The function returns a null-value if there is no Tasks in the specified Queue.

Otherwise, the function returns a dictionary containing the following elements:

id
>    the Task handle for the retrieved Task. In a [Cluster] environment, this Task and the current Task may be running on
>    different Cluster member computers

parameter
>    the value of the *parameter* parameter used when the Task was enqueued

**Note:** this function removes the first Task from the Queue. Unless the retrieved Task re-enqueues itself into the same Queue, no other Task will find it in that Queue.

# Formal Syntax

```
string              ::=  "string-data"
number              ::=  digits
name                ::=  alpha-numeric-and-underscore-starting-with-alpha

variable            ::=  name
codeName            ::=  name | name :: name

varDeclaration      ::=  var name 0*(, name)
varValueDeclaration ::=  var name [= expr] 0*(, name [= expr] )
dataRef             ::=  variable | expr [ expr ] | expr . name | expr . ( expr )
constDeclaration    ::=  const name = constExpr 0*(, name = constExpr)
constExpr           ::=  string | number | null | false | true
spawnExpr           ::=  spawn name [ ( expr ) ]
basicExpr           ::=  constExpr | dataRef | expr . name ( [argList] ) | spawnExpr

unaryOp             ::=  ! | not | - | +
unary               ::=  basicExpr | unaryOp unary | ( expr )
multOp              ::=  * | / | %
multBinary          ::=  unary | multBinary multOp unary
addOp               ::=  + | -
addBinary           ::=  multBinary | addBinary addOp multBinary
cmpOp               ::=  < | <= | == | != | >= | >
cmpBinary           ::=  addBinary | cmpBinary cmpOp addBinary
logicOp             ::=  & | and | | | or | && | and then | || | or else
logicBinary         ::=  cmpBinary | logicBinary logOp cmpBinary
ternary             ::=  logicBinary | logicBinary ? logicBinary : ternary
expr                ::=  ternary

argList             ::=  expr 0*(, expr)
funcCall            ::=  codeName ( [argList] )
procCall            ::=  codeName ( [argList] ) | expr . name ( [argList] )

letOp               ::=  = | += | -= | *= | /= | %= | |= | &=
letOper             ::=  dataRef letOp expr

nullOper            ::=  | null
stopOper            ::=  stop
returnOper          ::=  return [ expr ]

ifOper              ::=  if expr then opSequence 0*( elif expr then opSequence ) [ else opSequence ] end
                         [ if ]
altIfOper           ::=  if expr { opSequence } 0*( elif expr { opSequence } ) [ else { opSequence } ]

loopInitOp          ::=  letOp | varValueDeclaration
loopPreamble        ::=  while expr | for [ loopInitOp ] [while expr ] [ by letOp ]
altLoopPreamble     ::=  while expr | for ( [ loopInitOp ] ; [ expr ] ; [ letOp ] )
loopOper            ::=  [ loopPreamble ]loop opSequence 0*( exitif expr ; opSequence) end [ loop ]
```

```
altLoopOper         ::= altLoopPreamble { opSequence 0*( exitif expr ; opSequence) }

oper                ::= nullOper | procCall | letOper | returnOper | stopOper | ifOper | loopOper |
                        varValueDeclaration

altOper             ::= altIfOper | altLoopOper

seqOper             ::= oper ; | altOper

opSequence          ::= 0*( seqOper )


entryBody           ::= forward ; | is opSequence end [ entry ] ; | { opSequence }

procBody            ::= forward ; | external ; | is opSequence end [ procedure ] ; | { opSequence }

funcBody            ::= forward ; | external ; | is opSequence end [ function ] ; | { opSequence }

parmList            ::= name 0*(, name)

entry               ::= entry name entryBody

procedure           ::= procedure codeName ( [ paramlist] ) procBody

function            ::= function codeName ( [ paramlist] ) funcBody


program             ::= 1*(entry | procedure | function | varDeclaration | constDeclaration)
```

CommuniGate **Pro**

Applications  Data  CLI/API  Rules  CG/PL  **PBXApp**  XIMSS  WebApp  WSSP  Helpers

# Real-Time Applications

- **Application Environments**
- **Environment Files Hierarchy**
- **Managing Environments**
- **Application Model**
  - **Call Transfer**
  - **Bridged Calls**
  - **B2BUAs (Back-to-Back User Agents)**
- **CG/PL Applications**
  - **Input**
  - **Signals**
  - **Dialog**
  - **DTMF**
  - **Media**
  - **Bridges and Mixers**
  - **Call Transfer**
  - **Info Requests**
  - **Register Requests**
  - **Service Requests**
  - **Instant Messages and XMPP-type Requests**
- **Supported Media Formats**

The CommuniGate Pro Real-Time Application module provides an infrastructure to design and deploy applications processing various Signals.
Real-Time Applications used to process voice calls include voice mail, auto-attendant, conferencing, etc. These applications implement the "IP-PBX" functionality, providing an Internet standards-based alternative to legacy PBX (Private Branch Exchange) systems.

A Real-Time Application can be invoked when a Signal is sent to a certain Account, or certain Signals can be explicitly directed to Real-Time Applications.

The CommuniGate Pro Server software comes with several Real-Time Applications already built-in. These applications are highly customizable, and they can be used in most "regular" cases.

Read this section if you want to customize the built-in Applications, and/or if you want to create your own Real-Time Applications.

# Application Environments

A Real-Time Application Environment is a set of files that may include:

- CGPL files with Application code and code modules.
- Arbitrary service files - pre-recorded media files, for example.
- Language directories.

Environments are designed to support several human (spoken) languages. The default Environment language is English. To support other languages, an Environment should contain *language directories*, named as the languages they support (`french`, `russian`, `japanese`, etc.)

A language directory can contain the same files as the Environment itself, but it cannot contain language subdirectories.

Real-Time Application Tasks can select a language to be used. When a non-default language is selected, and the application code tries to read a file from the Application Environment, the selected language subdirectory is used. If the file is not found in the language subdirectory, the file is retrieved from the Application Environment itself (i.e. the default language file is used).

The CommuniGate Pro server comes with a built-in "Stock" Real-Time Application Environment. This Environment contains some basic applications and all files required by those applications, as well as some useful code section and media files.

Each CommuniGate Pro system has its Server-wide Application Environment. A CommuniGate Pro Dynamic Cluster installation also has a Cluster-wide Application Environment. To modify these Environments, open the WebAdmin Interface Domains section and follow the PBX link.

Each CommuniGate Pro Domain has its own Application Environment. To modify that Environment, open the WebAdmin Interface Domains section, open the Domain Settings for the selected Domain and follow the PBX link.

Modifications of the Cluster-wide Environment, as well as modifications of an Environment in any Shared Domain are automatically distributed to all Cluster Members.

Since Domains have their own Application Environments, different applications in different Domains can have the same name.

All Application Environment files should use the UTF-8 character set for non-ASCII symbols.

# Environment Files Hierarchy

Real-Time Application Tasks are executed "on behalf" of a certain CommuniGate Pro Account.
When an application requests an "environment file" and the default language is selected, the Server looks for the file in:

- the Account Domain Environment.
- the Server-wide Environment or (if the Account belongs to a Shared Domain) the Cluster-wide Environment.
- the Stock Environment.

If a non-default language is selected, the Server first looks for the file in the language directories of the Environments listed above. If the file is not found in any of those directories, the listed Environments themselves are searched. So, if a language-specific file has not been created, the default-language (English) file is used.

This hierarchy provides for simple Application customization. Accounts in all Domains can use the same Stock or Server-wide applications, while Domain Administrators can customize these applications by uploading custom files into their Domain Environments.

# Managing Environments

The WebAdmin Interface provides the Real-Time Application Environment Editor pages to manage Server-wide, Cluster-wide, and Domain Application Environments.

To manage the Server-wide and Cluster-wide Environments, open the Users realm of the WebAdmin Interface, and click the PBX link.

To manage the Domain Environment, open that Domain page in the Users realm of the WebAdmin Interface, and click the PBX link. The Domain Administrator should have the CanModifyPBXApps Access Right to be able to create and modify the Domain Application Environment.

The Environment Editor page contains the list of all files "visible" in this Environment: it lists files directly uploaded to this particular Environment, as well as all files uploaded to the Environments used as the "default files" source for this Environment:

| no file selected | | |
|---|---|---|
| **Name** | **Size** | **Modified** |

| | | | | |
|---|---|---|---|---|
| | Help.gif | | 155 | 25-Sep-04 |
| default | addressbook.sppi | | 5143 | 23-Sep-05 |
| default | voicemail.sppr | | 8K | 27-Feb-05 |
| default | failure.wav | | 10K | 27-Feb-05 |
| | ... | | | |
| default | xfer.wav | | 15K | 28-Sep-05 |
| | xonix.wav | | 30K | 02-Oct-05 |

Files directly uploaded to the Environment have a checkbox in the Marker column. Files from the other Environments "visible" in this Environment have the word `default` in that column.

You can download any of the Environment files by clicking the file name.

You can upload a file to the Environment by clicking the Browse button and selecting a file on your workstation, then clicking the Upload button.

You can delete any of the files uploaded to the Environment by selecting the checkboxes and clicking the Delete Marked button.

If you are uploading a file with the `.sppr` or the `.sppi` extension, the Editor assumes that the file contain some CG/PL program code, and it tries to compile that code.
If you are uploading a file with the `.settings` extension, the Editor assumes that the file contain a [dictionary](#), and it tries to parse it.
If the compiler or the parser detects an error, the file is not uploaded, and the file content is displayed on the Editor page, with the red `<--ERROR-->` marker indicating the location of the error.

The Server places used Real-Time Environment files into an internal cache. When you upload a file to any Environment, that Environment cache is automatically cleared. If you upload a file to a Shared Domain Environment or to the Cluster-wide Environment, the updated file automatically propagates to all Cluster Members.

You can upload a set of files by uploading a TAR-archive (a file with `.tar` name extension). For example, when you have a TAR-archive with a predesigned Real-Time Application you can open the Environment you want to modify, and upload the `.tar` file. The Server will unpack the archive and store each file individually, as if they were uploaded one-by-one.

The Editor page contains the list of all Language directories:

| Languages |
|---|
| arabic |
| french |
| russian |
| |

To create a new Variant, enter the language name, and click the Create Language button.

To open a Language directory, click its name. The Editor will display the Language name, and it will provide the UP link to the list of the default language files.

# CLI/API

The Command Line Interface/API can be used to manage Application Environments. See the [Real-Time Application Administration](#) section for the details.

# Virtual File Storage Area

The Application Environments can be modified using [FTP](#) and [HTTP](#) clients, [CG/PL](#) applications, and any other method that provides access to the Account [File Storage](#).
Special `$DomainPXBApp`, `$ServerPXBApp`, and `$ClusterPXBApp` directories provide access to the Domain and Server/Cluster-wide

# Application Model

Real-Time Applications run as [Tasks](). To start an Application, the CommuniGate Pro Server starts a new Task and instructs it to execute the `main` [entry]() code section of the specified Application.

A Real-Time Application Task can run in the *disconnected* mode, or it can be a part of exactly one Real-Time *session* (such as a phone call) with some *peer*. A *peer* is any Real-Time entity, such as a SIP phone, a PSTN gateway, a remote Real-Time application communicating via SIP, or a local Real-Time Application, i.e. some other Task.

If a Task is participating in a session with some *peer*, it can be in one of the following modes:

`incoming`
> an incoming session (a *call* in the telephony terms, an `INVITE` *request* in the SIP terms) has been directed to the Task, but the Task has not accepted the session (call) yet.

`provisioned`
> an incoming session has been directed to the Task, the Task has not accepted the session yet, but it has sent a provisional response to the caller.

`connecting`
> the Task has initiated an outgoing session (call), but the session has not been established yet.

`connected`
> the Task has accepted an incoming session, or the Task has established an outgoing session.
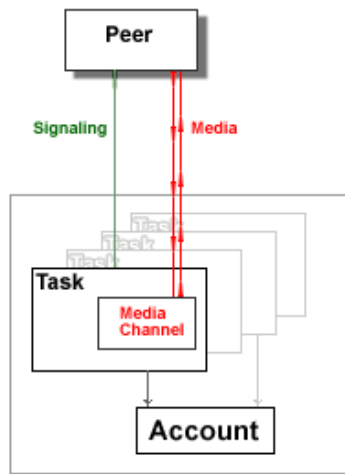


A Task can receive [Signals]() from its peer, and it can send Signals itself. Signals can be used to end the current session, to update the session parameters, etc. Some of the Signals sent by the peer are processed by the Real-Time Application Environment itself, while other Signals are passed to the Task for processing.

A Real-Time Application Task can have a Media Channel associated with it. When a session is established, the Task Media Channel is connected to the peer's media channel.

A Task can use its Media Channel to send media (audio, video) to the peer, and to record media sent by the peer.

A Task can switch the peer media somewhere else - for example, to a separate service providing *music on hold* or other media services. When the peer media is switched away, the Task Media Channel cannot be used, but the Task still can control the peer by sending Signals to it and it still receives Signals from the peer. The Task can switch the peer media back to its own Media Channel.
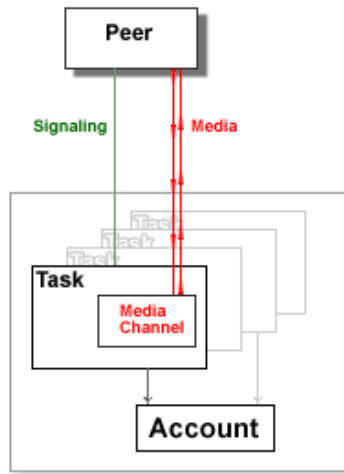
A Media Channel provides a *conversation space*. A Task can attach other peer media to the Task own Media Channel. This operation creates a conversation space (or a *conference*) that includes the Task own peer and all peers with media attached to this Task. Media sent by any peer in a *conversation space* is relayed to all other peers in that space, using the data mixing algorithms and parameters defined for that media type.
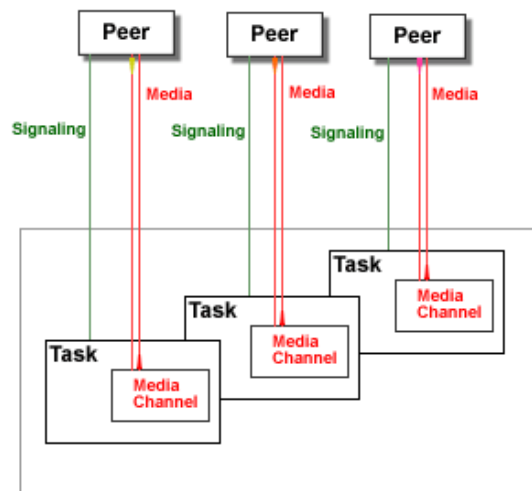


A Task with attached peer media can use its Media Channel to send media to its own peer and to all attached peers at once.

## Call Transfer

Real-Time Application Tasks can automatically process Call Transfer requests.

When a Task receives a REFER request (from a remote client, or from a different Task), it can:

- process it automatically (this is the default option)
- reject it
- pass it to the application

If a Task is instructed to processes REFER requests automatically, it creates a new INVITE request and sends it to the address specified in the Refer-To: field. While this INVITE request is in progress, the task informs the peer about the call progress by sending it NOTIFY requests.

If the INVITE request is completed successfully, the Task sends the BYE Signal request to the current peer (unless it has already received the BYE Signal from the current peer), and then it switches its Signal dialog and the Media Channel to the new peer:
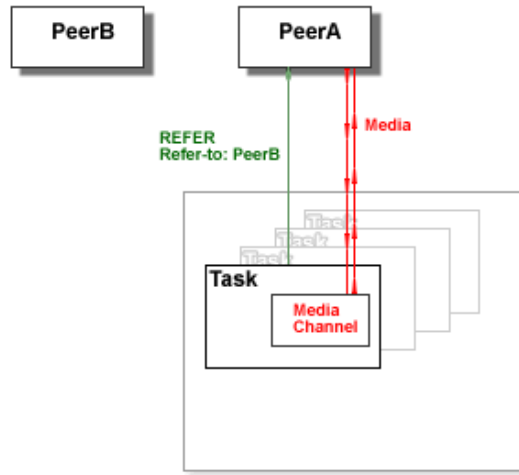
The INVITE signal initiated with the REFER signal can contain authentication information. The application program can instruct the Task:

- to authenticate the INVITE signal as coming from the current Account. This is the default option.
- to authenticate the INVITE signal as coming from the peer (the entity that has sent the REFER signal).
- to reject REFER signals.

To process "Assisted" Call Transfer, the Real-Time Application engine detects all INVITE signals with the Replaces field. If the dialog data specified in the Replaces field matches an active dialog established in any Real-Time Task, the INVITE signal is directed to that Task. The Task sends a BYE signal to the current peer, and it switches its Signal dialog to the source of this INVITE signal, and it switches the Media channel to the entity specified in the INVITE signal.
This processing takes place automatically, transparently to the application program the Task is running.
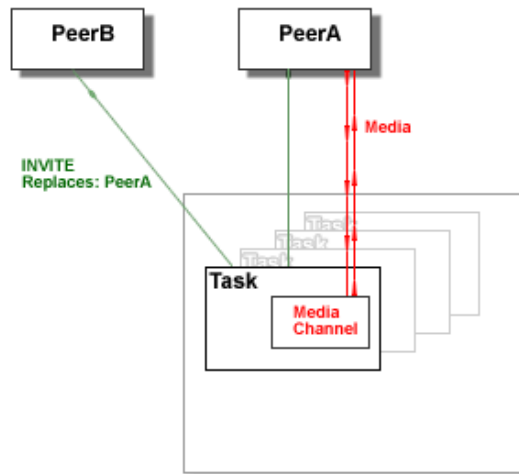


## Bridged Calls

A pair of Real-Time Application Tasks can build a Media Bridge. When a Media Bridge is built, the Tasks' peers exchange media, while each Task stays in control of the peer signaling.

To build a bridge:

- an application program some Task A is running uses the StartBridge CG/PL function to send a special `[bridgeStart]` event to some other Task B. The request contains the the current Task A peer media description.
- the application program the Task B is running receives this request.
- the Task B application program uses the AcceptBridge CG/PL function to build the bridge.
- the Real-Time Application engine sends a re-INVITE signal to the Task B peer, switching its Media to the Task A peer Media.

- the Real-Time Application engine sends a `[bridgeResponse]` event to Task A. This event contains the Task B peer media description.
- the AcceptBridge operation in Task B application program completes, and the Task B application program continues to run.
- when the Task A receives the `[bridgeResponse]` event, the Real-Time Application engine processes the event itself, and it does not deliver it to the Task A application program.
- the Real-Time Application engine sends a re-INVITE to the Task A peer, switching its Media to the Task B peer Media.
- the StartBridge operation in Task A completes, and the Task A application program continues to run.
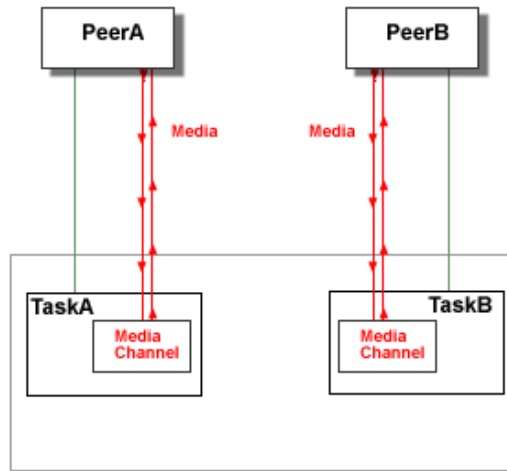


When the Tasks A and B are in the "bridged" state, one of the Tasks (Task B here) can receive a re-INVITE signal from its peer. The new media description in this signal request may ask the Task to change its media parameters (hold/resume, switching to a different media source, etc.). This re-INVITE signal is processed automatically, without any involvement of the application programs the Tasks are running:

- the Real-Time Application engine sends a special `[bridgeUpdate]` event to the Task A. This event contains the new media description.
- the Real-Time Application engine processes the `[bridgeUpdate]` event itself, and it does not deliver it to the Application program the Task A is running.
- the Real-Time Application engine sends a re-INVITE signal to the Task A peer, switching it to the new Media source of the Task B peer.
- the Real-Time Application engine sends the new Peer A media description (received with the re-INVITE response) to the Task B as a special `[bridgeResponse]` event.
- when the Task B receives the `[bridgeResponse]` event, the Real-Time Application engine process it itself and does not deliver it to the Task B application program.
- the new Task A media descriptor received is sent to the Peer B as a response to the initial re-INVITE signal, switching the Task B peer to the new Task A peer media.
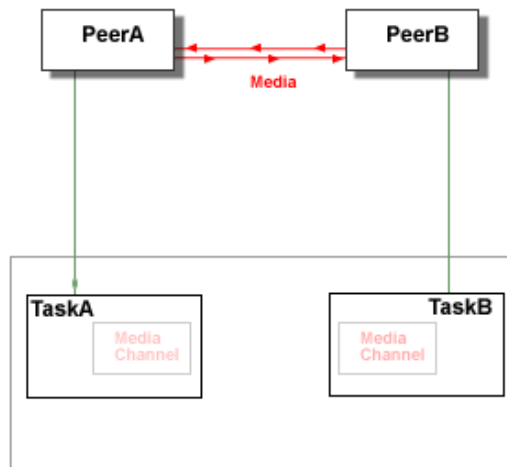


When the Tasks A and B are in the "bridged" state, one of the Tasks (Task A here) can receive a Call Transfer (REFER) signal

from its peer:

- the the Real-Time Application engine composes the INVITE signal for the Task A, and sends it to the referred peer; the Signal contains the Task B peer media description.
- the INVITE signal succeeds and the new Task A peer (peer C) media description is received.
- the Real-Time Application engine automatically sends a special `[bridgeUpdate]` event to the Task B, with new peer media description.
- when the Task B received the `[bridgeUpdate]` event, the Real-Time Application engine processes the event itself, and it does not deliver it to the Task B application program.
- the Real-Time Application engine sends a re-INVITE signal to the Task B peer, switching it to the Media source of the new Task A peer.
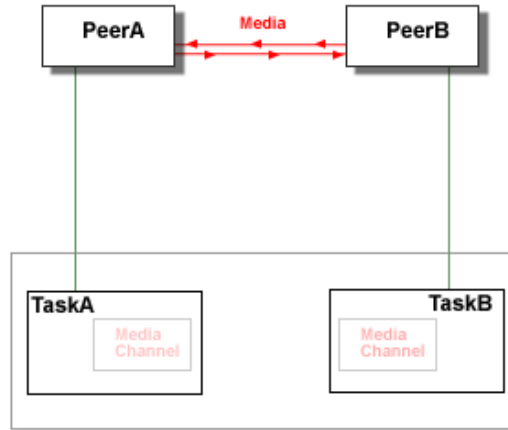


When the Tasks A and B are in the "bridged" state, one of the Tasks (Task A here) can receive Call Transfer (INVITE with Replaces) signal (delivered to it by the [Signal](#) module based on the Replaces field content):

- the Real-Time Application engine automatically sends a special `[bridgeUpdate]` event to the Task B, with new Task A peer media parameters.
- the Real-Time Application engine sends a re-INVITE signal to the Task B peer, switching it to the Media source of the Task A new peer.
- the Real-Time Application engine sends the INVITE response to the new Task A peer with the Task B peer media parameters, connecting the new Task A peer and the Task B peer media.

When the Tasks A and B are in the "bridged" state, one of the Tasks (Task A here) can decide to break the Media Bridge:

- an application program the Task A is running uses the [BreakBridge](#) CG/PL function to break the Media bridge.
- the Real-Time Application engine automatically sends a special `[bridgeClose]` event to the Task B, and sends a re-INVITE signal to the Task A peer, switching it to the Task A Media Channel.
- when the Task B receives the `[bridgeClose]` event, the Real-Time Application engine automatically sends a re-INVITE request to the Task B peer switching it to the Task B Media Channel.
- the Real-Time Application engine delivers the `[bridgeClose]` event to the Task B application program.
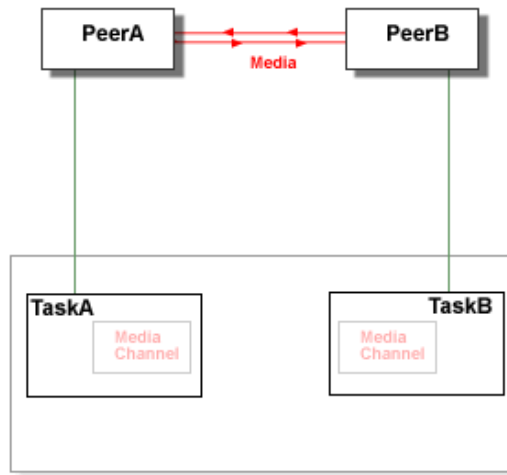
When the Tasks A and B are in the "bridged" state, one of the Tasks (Task A here) can receive a disconnect (BYE) signal from its peer, or it can quit for various reasons. In this case:

- the Real-Time Application engine automatically sends a special `[bridgeClose]` event to the Task B.
- when the Task B receives the `[bridgeClose]` event, the Real-Time Application engine automatically sends a re-INVITE request to the Task B peer switching it to the Task B Media Channel.
- the Real-Time Application engine delivers the `[bridgeClose]` event to the Task B application program.
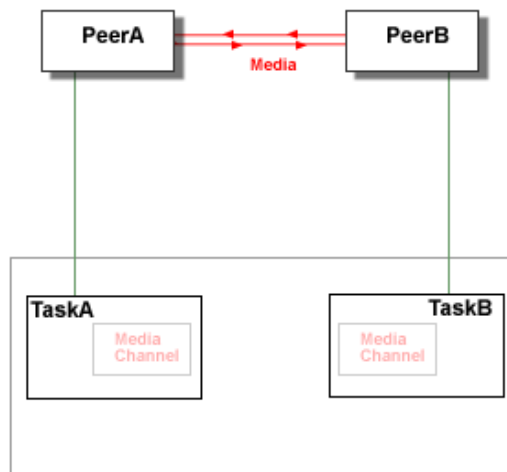


The Real-Time Application engine may choose to implement a Media Bridge using one Task Media Channel as a "media relay".

## B2BUAs (Back-to-Back User Agents)

The Bridged Call functionality is used to implement the B2BUA (Back-to-Back User Agents) technology. A pair of Real-Time Tasks in the Bridged Mode can work as a "smart proxy": while peers connected to these tasks can communicate directly, the signaling is still controlled with the Tasks and the application programs these Tasks are running.

---

# CG/PL Applications

Real-Time Applications can be written using the CG/PL language.

When a Task is created to process an incoming call, the `main` entry of the specified CG/PL application program is executed.

Real-Time Applications can use CG/PL external-declarations. When a code section (a procedure or a function) declared as *external* is called, a file with the code section name and the `.sppi` extension is loaded from the current Environment. The program code in this file must contain the code section with the specified name and of the proper type (a procedure or a function).

The program code in an `.sppi` file may contain other code sections as well.

Real-Time Applications can use the following built-in procedures and functions.

## Input

`ReadInput(timeOut)`

This function is used to receive external Task communications: DTMF symbol entered by the peer, signals sent by the peer, and Events sent by other Tasks and by the system itself. See the CG/PL Events section for the detail.
The *timeOut* value should be a number specifying the maximum wait period (in seconds). If the *timeOut* value is zero, the function checks for pending digits and events, without any waiting.
The function returns:

- a string with the first DTMF symbol in the Task DTMF buffer. The symbol is removed from the Task buffer.
- a dictionary with the first waiting Event. The Event is removed from the Task Event queue.
- a null-value if no DTMF symbol and no Event was received during the specified time period.

When the peer disconnects, the Task receives a Disconnect Event from the system (this Event dictionary does not contain the `.sender` element).

`IsDisconnectEvent(input)`

This function returns a true-value if the *input* value is a Disconnect Event.

```
//
// Sample: ReadInput()
//   Accept an incoming call (stop if it's not possible).
//   Play the PressPound media file.
//   Wait for any input for up to 5 seconds.
//   If the "pound" ("#") symbol was entered,
//     play the Good media file.
//   Otherwise,
//     play the Bad media file.
//   Stop.
//
entry Main is
  if AcceptCall() != null then stop; end if;
  PlayFile("PressPound");
  PlayFile(ReadInput(5) == "#" ? "Good" : "Bad");
end entry;
```

## Signals

`AcceptCall()`

This function accepts an incoming session, if there is one: the Task should be in the `incoming` or `provisioned` mode.
This function returns a null-value if the session is accepted successfully, and the Task is placed into the `connected` mode.
If a session cannot be accepted, this function returns an error code string, and the Task is placed into the `disconnected` mode.

```
//
// Sample: AcceptCall()/RejectCall()
//   If the current local time is not between 8:00 and 17:00,
//     reject the call (with the 403 error code) and stop.
//   Otherwise,
//     accept the call (stop if it is not possible)
//     play the Welcome media file and stop.
//
entry Main is
  currentTime = TimeOfDay(GMTToLocal(GMTTime()));
  currentHour = currentTime / 3600;
  if currentHour < 8 or currentHour >= 17 then
    RejectCall(403); stop;
  end if;
  if AcceptCall() != null then stop; end if;
  PlayFile("Welcome");
end entry;
```

RedirectCall(*newURI*)
> This procedure redirects an incoming session, if there is one: the Task should be in the `incoming` or `provisioned` mode. The *newURI* value should be a string, or an array of strings. The incoming session is redirected to the URI(s) specified and the Task is placed into the `disconnected` mode.

ForkCall(*newURI*)
> This procedure redirects an incoming session, if there is one: the Task should be in the `incoming` or `provisioned` mode. The *newURI* value should be a string, or an array of strings. The incoming session is directed to the URI(s) specified, and the current Task remains in the same state, so it can accept, reject, redirect, provision, or fork this call later.

ProvisionCall(*media*,*reliably*)
> This function sends a provisional Response for an incoming session Request, if there is a pending one: the Task should be in the `incoming` or `provisioned` mode.
>
> If the *media* value is not a null-value, then a Task Media Channel is created, the Task is placed into the `provisioned` mode and the Media Channel operations (such as `PlayFile`) can be used to generate "ring-back tones".
>
> If the *reliably* value is not a null-value, the response confirmation (SIP `PRACK`) is requested, and the Task is suspended till a confirmation request arrives.
>
> This function returns a null-value if the response is sent successfully.
>
> If a provisional response cannot be sent, this function returns an error code string.

ProvisionCall(*parameters*)
> This function is an extension of the `ProvisionCall` function shown above. The function parameters are specified using the *parameters* dictionary, which can contain the following elements:
>
> media
>> if this element is present, then it has the same effect as a non-null value of the *media* parameter of the ProvisionCall function.
>
> reliably
>> if this element is present, then it has the same effect as a non-null value of the *reliably* parameter of the ProvisionCall function.
>
> responseCode
>> if this element is present, its value should be a number in the 101..199 range (inclusive). It specifies the numeric Response code sent. If this element is not present, the 180 value is used.
>
> responseText
>> if this element is present, its value should be a string. It specifies the textual Response code sent. If this element is not present, the default Response text is used.
>
> reasonCode
>> if this element is present, its value should be a positive number. The Reason header field is added to the Response, using this element and the `reasonText` element values.
>> If the element value is less than `1000`, the composed field contains `SIP` as the "protocol" value, otherwise the protocol value is set to `Q.850`, and 1000 is substracted from the element value.
>
> reasonText
>> if this element is present, its value should be a string. This element is used only when the `reasonCode` element is a positive number.

RejectCall(*responseCode*)
> This procedure rejects an incoming session if there is one: the Task should be in the `incoming` or `provisioned` mode. The *responseCode* value should be a numeric value between 400 and 699, or an error code string. This number is sent back to the caller as the Signal Response code. If the code 401 is sent back, and the request came from outside the CommuniGate Pro Server (via the [SIP](#) protocol), the SIP server module adds the proper fields to the response to facilitate client Authentication.
>
> Alternatively, the *responseCode* value can be a dictionary with the optional `responseCode`, `responseText`, `reasonCode`, `reasonText` elements, which have the same meanings as the elements of the `ProvisionCall()` function parameter. The Task is placed into the `disconnected` mode.

```
//
// Sample: RedirectCall()/ProvisionCall()
//   Provision the call (stop if it is not possible).
//   If the current local time is between 12:00 and 13:00,
//     fork the call to user1 in the same domain.
//   Play the "PleaseWait" media file.
//   If the current local time is not between 8:00 and 17:00,
//     redirect the call to user2 in the same domain, and stop.
//   Otherwise,
//   Accept the call (stop if it's not possible).
//   Play the Welcome media file, and stop.
//
entry Main is
  if ProvisionCall(true,true) != null then stop; end if;
  currentTime = TimeOfDay(GMTToLocal(GMTTime()));
  currentHour = currentTime / 3600;
  if currentHour >= 12 and currentHour <= 13 then
    ForkCall("sip:user1@" + MyDomain());
    stop;
  end if;
  PlayFile("PleaseWait");
  if currentHour < 8 or currentHour >= 17 then
    RedirectCall("sip:user2@" + MyDomain());
    stop;
  end if;
  if AcceptCall() != null then stop; end if;
  PlayFile("Welcome");
end entry;
```

**Note:** if a pending incoming call has been canceled, the Task receives a Disconnect Event, and the Task mode changes to *disconnected*.

SetCallParameters(*parameters*)

This procedure sets call (INVITE dialog) parameters.

The *parameters* value should be a dictionary with new call option parameters. The existing parameter values are removed. The following parameters are used for all dialog requests:

Max-Forwards

a positive number used to limit the number of "hops" a request can traverse.

customIdentity

a string or a dictionary to be used as the request P-Asserted-Identity header field. This field is also added to the provisioning responses.

Privacy

a string or an array of strings to be used as the request Privacy header field.

IPSource

the IP Address object that overrides the Server rules and explicitly sets the source IP Address for outgoing SIP packets and locally generated media.

allowedAudioCodecs

if this parameter is an array, it lists the names of audio codecs that can be included into SDP documents sent with this task.

allowedVideoCodecs

if this parameter is an array, it lists the names of video codecs that can be included into SDP documents sent with this task. If this array is empty, the sent SDP does not include the "video" media, unless this SDP is an answer to an SDP offer that includes the "video" media.

allowedSSRC

if this parameter is "NO", all "ssrc" attributes are removed from SDP.

allowedAttributes

if this parameter is "nonCustom", all custom attributes (with names starting with x-) are removed from SDP.
if this parameter is "required", all attributes except the required ones are removed from SDP. The "phone", "info", "uri", "email", "time", "zone", "key", "bandwidth", "title" SDP elements are removed, too.

The following parameters are used when a call dialog is established:

Session-Expires
> a non-negative number specifying the default session expiration (dialog refreshing) time period. If set to zero, no dialog refreshing takes place.

Some parameters are used when a call is initiated (with the `StartCall`, `StartBridgedCall` functions) or when a call is accepted (with the `AcceptCall` function) - see below.

StartCall(*destination*)
> This function initiates an outgoing session. The Task should be in the `disconnected` mode.
> The *destination* value should be a string containing the URI to send a session request to, or a dictionary containing the following string elements:

`""` (empty string)
> the URI to send the request to.

From (optional)
> an E-mail, URI, or field-data to use for the request `From` field. If not specified, the current Account data is used.

To (optional)
> an E-mail, URI, or field-data to use for the request `To` field. If not specified, the request URI is used.

Call-ID (optional)
> a string to use for the request `Call-ID` field. If not specified, a new Call-ID is generated.

The following optional parameters are taken from the *destination* dictionary. If they are absent (or the *destination* is not a dictionary), they are taken from the current call parameters set with the `SetCallParameters` procedure:

authUsername, authPassword
> credentials (the authenticated name and password strings) to be used if an outgoing request is rejected with the 401 error code.

Expires
> a number specifying the maximum calling (alerting) time (in seconds).

Subject
> request Subject string.

Remote-Party-Id
> an array of dictionaries, each used to compose request `Remote-Party-Id` fields.

Privacy
> a string or an array of strings to compose the request `Privacy` field.

Diversion
> an array to compose the request `Diversion` fields.

P-CGP-Private
> a string to compose the request `P-CGP-Private` field. This field can be used to pass arbitrary parameters between sending and receiving Tasks (on the same or different systems).

P-CGP-Local
> a string to compose the request `P-CGP-Local` field. This field can be used to pass arbitrary parameters between sending and receiving Tasks (only within the same single-server or Cluster systems).

P-Billing-Id
> a string to compose the request `P-Billing-Id` field stored in the Call object. This string is added to the server-generated CDRs.

Call-Info
> an array to compose the request `Call-Info` fields.

Alert-Info

a string to compose the request `Alert-Info` field.

**Replaces**

a string or a dictionary to compose the request `Replaces` field. If a dictionary is used, it should contain the string values `Call-ID`, `fromTag`, `toTag`, and a boolean value `early-only`.

**Via**

if this string parameter is specified, it should contain a URI. The URI is added to the request as its Route field, forcing the request to be sent via that URI.

**NoInitSDP**

if this parameter is specified, the call establishment request does not contain the local Media channel SDP.

**No100rel**

if this parameter is specified, the outgoing requests do not have the "100rel" SIP option advertised as supported.

**mediaRelay**

if this parameter exists, a Media Proxy is always created for an outgoing call.

**TimerB**

if this parameter exists, its numeric value overrides the default "Timer B" value for an outgoing SIP transaction: this is the number of seconds to wait till the remote site sends any response (such as 100-Trying).

**Relay**

if this parameter exists and its value is `"NO"`, the outgoing SIP request cannot be relayed to non-client destinations.

**encryptMedia**

if this parameter exists and its value is `"NO"`, SDP documents (offers) sent with this task will not include elements needed to enabled media encryption.

This function returns an error code string if an outgoing call cannot be initiated.
If an outgoing call has been initiated successfully, the function returns a null-value, and the Task starts to receive call progress Events from the system: zero or more *provisional response* Events, followed by exactly one *final response* Event. If the outgoing session has been established successfully, the Task receives a final response Event without a parameter. If the outgoing session has not been established, the Task receives a final response Event with a parameter - the error code string.

`IsCallProvisionEvent(`*input*`)`

This function returns a true-value if the *input* value is a call provisional response Event. Otherwise the function returns a null-value.

`IsCallCompletedEvent(`*input*`)`

This function returns a true-value if the *input* value is a call final response Event. Otherwise the function returns a null-value. When a Task receives a final response Event, the call signaling process has completed. If the Event has a parameter, the call has failed, and the parameter contains the error code string.

`CancelCall()`

If a Task has a pending outgoing call (initiated using the StartCall function), this procedure cancels that call (the Task will not receive a final response Event).

`Disconnect()`
`Disconnect(`*reasonCode*`,`*reasonText*`)`

This procedure ends the active session, if any, and the Task is placed into the `disconnected` mode.
If the *reasonCode* is specified, its value should be a positive number, and if the *reasonText* is specified, its value should be a string. If the BYE signal is sent to the peer, the Reason header field with the *reasonCode* and *reasonText* values is added to the signal request.

```
//
// Sample: StartCall()/Disconnect()
//    Accept an incoming call (stop if it's not possible).
//    Remember the caller URI.
//    Play the CallBack media file.
//    Disconnect();
//    Call the caller back (stop if it's not possible).
//    Play the IamBack media file, and stop.
```

```
//
entry Main is
  if AcceptCall() != null then stop; end if;
  fromWhom = RemoteURI();
  PlayFile("CallBack");
  Disconnect();

  if StartCall(fromWhom) != null then stop; end if;

  loop
    input = ReadInput(3600);
  exitif not IsCallProvisionEvent(input);
    null;
  end loop;

  if not IsCallCompletedEvent(input) or else
     input.parameter != null then stop; end if;

  PlayFile("IamBack");
end entry;
```

`IsConnected()`
>   This function returns a true-value if the Task is in the `connected` mode.

`IsHalfConnected()`
>   This function returns a true-value if the Task is in the `connected` or `provisioned` mode.

# Dialog

`RemoteURI()`
>   This function returns a string with the peer URI (taken from the dialog From/To addresses). If there is no session in place, the function returns a null-value.

`LocalURI()`
>   This function returns a string with the Task URI.

`IncomingRequestURI()`
>   This function returns a string with URI of the pending incoming INVITE request. If there is no pending incoming INVITE request, the function returns a null-value.

`RouteLocalURI(`*uri*`)`
>   This function tries to [Route] the E-mail address from the specified URI. If the URI cannot be parsed, or the URI address cannot be routed, or it routes to a non-local address (i.e an E-mail address hosted on a different system), this function returns a null-value. Otherwise, the function returns the E-mail address of the CommuniGate Pro user the original URI address is routed to.
>   This function allows you to correctly process all [Forwarders], [Aliases], and other CommuniGate Pro Routing methods.

`RemoteIPAddress()`
>   This function returns an ip-address object the session establishment request was received from or was sent to. This IP Address/port pair is the actual peer address or the address of a proxy used to relay the peer signaling.

`RemoteAuthentication()`
>   This function returns a null-value if the session starting request was not authenticated, or a string with the authenticated user E-mail address.

`PendingRequestData(`*fieldName*`)`
>   This function returns a data element of the pending incoming request.
>   If a request is pending, the function returns the following data, depending on the *fieldName* value, which should be a string:
>
>   `Call-ID`
>   >   the function returns a string with the request Call-ID value.
>
>   `From, To, Referred-By`
>   >   the function returns a dictionary if the field exists in the request. The dictionary contains the following elements:
>   >
>   >   `""` (and element with empty string key)
>   >   >   the address (in the *username@domain* form)
>   >
>   >   `@realName`

a string with the "display-name" part of the address

      `@schema`
          a string with the URI schema (if absent, `sip` is assumed)

      `@port`
          a number with the "port" part of the URI

      `@tag`
          the "tag" field parameter

      `@field-params`
          a dictionary with other field parameters.

      `@headers`
          a dictionary with URI headers.

      *anyOtherName*
          a URI parameter.

All elements except the address element are optional.

`Remote-Party-Id`, `History-Info`
the function returns an array if the field or fields exist in the request. Each array element is a dictionary with the same elements as in the `From` field dictionary.

`Foreign-Asserted-Identity`
the function returns an array of the request `P-Asserted-Identity` fields. Each dictionary contains the same elements as the `From` field dictionary.

`Route`, `Record-Route`, `Diversion`, `Via`, `Path`, `Supported`, `Require`, `Proxy-Require`, `Privacy`, `Allow`, `Allow-Events`
the function returns an array containing one or more strings with field values. If no field value exists, the function returns a null-value.

`CSeq`
the function returns a number - the value of the CSeq field numeric part.

`Max-Forwards`
the function returns a number - the value of the Max-Forwards field.

`User-Agent`, `Reason`, `P-CGP-Private`, `P-CGP-Local`
the function returns a string - the field value. If the field is absent, the function returns a null-value.

`Accept`
the function returns an array containing 2 strings for each field value: the accepted content type and the accepted content subtype. If the field is absent, the function returns a null-value.

`xmlsdp`
the function returns an [XML presentation](#) of the pending request SDP.

If no request is pending, the function returns a null-value.

`PendingRequestExData(`*fieldName*`)`
This function returns a non-standard data element of the pending incoming request.
The *fieldName* value should be a string. It specifies the name of a non-standard request field.
If a request is pending, the function returns specified field data.
If no request is pending, the function returns a null-value.

`SetLocalContactParameter(`*paramName*`,`*paramValue*`)`
This procedure allows you to add "field" parameters to the Contact field data for this Task dialog(s).
The *paramName* value should be a string. It specifies the field parameter name.
If the *paramValue* value is a non-empty string, it specifies the field parameter value.
If the *paramValue* value is an empty string, a parameter without a value is set (such as `isfocus`).

If the *paramValue* value is a null-value, a previously set field parameter is removed.

## DTMF

Each Task has a DTMF buffer string. When a DTMF symbol is received either in an INFO Request or as a media packet (via the Media Channel), the symbol is appended to this buffer.

`DTMF()`
> This function returns a string with the current content of the DTMF buffer. The DTMF buffer is not changed. Usually this function is not used, the `ReadInput()` function is used instead.

`ClearDTMF()`
> This procedure empties the DTMF buffer.

`SetInterruptOnDTMF(arg)`
> This function sets a flag controlling media playing and recording interrupts with received DTMF symbols.
> The *arg* value specifies the new flag value: if this value is a null-value, received DTMF symbols will not interrupt media playing or recording, otherwise received DTMF symbols will interrupt media playing and recording.
> When a Task is created, this flag value is set to a true-value.
> The function returns the previous value of this flag.

`SendDTMF(symbol)`
> This function sends a DTMF symbol to the peer.
> The *symbol* value should be a string containing 1 DTMF symbol.
> The function returns a null-value if the symbol has been sent, or a string with an error code if sending failed.

```
//
// Sample: ReadInput()/SendDTMF()
//   Accept an incoming call (stop if it's not possible).
//   Wait for an input for 10 seconds. If no input - stop.
//   If a digit is entered
//     play that digit, and send it back.
//     (using "0" ... "9" files)
//   If a star ("*") is entered,
//     wait for a digit (for 3 seconds)
//     and play the digit number square (2 digits)
//   If a pound ("#") is entered or the Peer
//     has disconnected, or any Event was sent, stop.
//
entry Main is
  if AcceptCall() != null then stop; end if;
  loop
    input = ReadInput(10);
       if input == "*" then
         input = ReadInput(3);
         if IsString(input) and input != "#" then
           input = "*" + input;
         end if;
       end if;
  exitif not IsString(input) or input == "#";
    if Substring(input,0,1) != "*" then
         PlayFile(input);
         void(SendDTMF(input));
       else
         input = Number(Substring(input,1,1);
         product = input *  input;
         PlayFile(String(product/10));
         PlayFile(String(product%10));
       end if;
  end loop;
end entry;
```

## Media

`PlayFile(fileName)`
`PlayFile(fileName,msec)`
> This procedure retrieves a file from its Application Environment and plays it. The string parameter *fileName* specifies the file name. If the specified file name does not contain a file name extension, the supported extensions are added and tried. The file should contain media data in one of the supported formats.
> If the *msec* parameter is specified, its value should be a number. Then the specified file is played for *msec* milliseconds,

repeating file in a loop if the file media playing period is shorter than the specified period. If the *msec* parameter has a negative value, the file is played in an infinite loop.

Playing is suppressed or interrupted if the session ends, if the DTMF buffer is not empty (unless DTMF Interruption is disabled), or if there is an Event enqueued for this Task.

```
Play(waveData)
Play(waveData,msec)
```
This procedure plays the *waveData* value which should be a datablock.

The datablock should contain media data in one of the [supported formats](#).

If the *msec* parameter is specified, it works in the same way as it works for the PlayFile procedure.

Playing is suppressed or interrupted if the session ends, if the DTMF buffer is not empty (unless DTMF Interruption is disabled), or if there is an Event enqueued for this Task.

```
PlayTone(freq,msec)
PlayTone(freq,msec,freq2)
PlayTone(freq,msec,freq2,ratio)
```
This procedure plays a "tone" (a sine wave). The *freq* parameter value should be a non-negative number - the wave frequency (in Hz). If the value is zero, audio silence is generated.

The *msec* parameter value should be a number, it works in the way as it works for the PlayFile procedure.

If the *freq2* parameter is specified, the generated tone is a combination of two sine waves, and the parameter value should be a positive number specifying the second wave frequency (in Hz).

If the *ratio* parameter is specified, its value should be a positive number in the 1..10000 range. It specifies the relative amplitude of the second wave, the first wave amplitude being 100.

Playing is suppressed or interrupted if the session ends, if the DTMF buffer is not empty (unless DTMF Interruption is disabled), or if there is an Event enqueued for this Task.

```
GetPlayPosition()
```
This function returns a number - the portion of the media (in milliseconds) played by the last `PlayFile` or `Play` operation. If the media was played in a loop, the length of the played portion in the last "loop" is returned: if a 3-second sound was played in loop, and it was interrupted after 7.2 seconds, the function returns 1200.

If the last Play* operation did not play its sound (because the DTMF buffer was not empty or there was an enqueued Event), the function returns either 0, or a negative value.

```
SetPlayPosition(arg)
```
This procedure instructs the very next `Play*` operation to start playing its sound not from its beginning.

The *arg* value should be a non-negative number. It specifies how many milliseconds of the media should be skipped before the Play* operation starts to play the media.

```
IsPlayCompleted()
```
This function returns a true-value if the media played by the last `Play*` operation was played in full. If media playing was interrupted, the function returns a null-value.

```
Record(timeLimit)
```
This function records incoming audio data. The *timeLimit* value should be a positive number, it specifies the maximum recording time in seconds.

Recording is suppressed or interrupted if the session ends, if the DTMF buffer is not empty, or if there is an Event enqueued for this Task.

The function returns a null-value if recording was suppressed, or a datablock with the recorded sound in the `WAV` format.

```
SetLocalHold(flag)
```
This procedure sets the current call "on Hold" (if the *flag* is a true-value), or releases the call from hold (if the *flag* is a null-value).

This procedure can be used only when the Task is in the *connected* mode, and the Task peer media is not bridged.

```
ReleaseMediaChannel()
```
This procedure releases the Task Media Channel (releasing the associated system resources).

This procedure can be used only when the Task is in the *disconnected* mode, or when the Task peer media is bridged.

```
MediaOption(optionName)
MediaOption(optionName,newValue)
```
This function return the current value of a Task Media Channel option.

If the *newValue* is specified and its value is not a null-value, then it is set as the new option value.

The *optionName* value should be a string and it specifies the option name. The following options are supported:

"preplay"

this a numeric option, is specifies the amount of pre-recorded media (in milliseconds) sent to remote peers "ahead of time", to pre-fill the peer jitter buffers.

"mixerDelay"

this a numeric option, it specifies the delay (in milliseconds) the Media Channel introduces before it reads Media Leg audio and mixes it with other Media Leg audio when a Media Channel has more than 2 active Media Legs ("conferencing"). The larger the delay, the better the Media Channel mixer processes irregular media streams coming from conference participants connected via sluggish networks.

"inputSilenceLevel"

this a numeric option, it specifies the minimum level of the incoming audio. If the level is lower than the specified value, it is processed as a complete silence - it is not recorded and if there are more than 2 Media Legs ("conferencing"), this Media Leg incoming audio is not used mixed.

"skipSilence"

this a numeric option, it specifies a time limit (in milliseconds). When the Media Channel is used to record the incoming media and the incoming audio level is lower than the "silence level" (see above), and this silence lasts longer than the specified time limit, recording is suspended. It resumes as soon as the incoming audio level increases above the "silence level".

"stopRecordOnPause"

this a numeric option, it controls recording of silence. When the Media Channel is used to record the incoming media and the incoming audio level is lower than the "silence level" (see above), and this silence lasts longer than the time limit specified with the "skipSilence" option (see above), recording is stopped if the value if this option is set to 1. The Task also receives special DTMF signal #15.

"mixMOH"

this a numeric option, it specifies if media from a peer sending MOH (Music-on-hold) should be processed or it should be ingored. If the value is 1, the MOH media is processed, if the value is 0 (default), the MOH media is ignored.
If there are not more than two peers connected to the Media Channel, MOH media is always processed.

```
//
//  Sample: Record()/PlayFile()/Play()
//    Accept an incoming call (stop if it's not possible).
//    Play the GreetingIs file.
//    Read the current prompt from
//       the MyGreeting.wav file in the File Storage.
//    Loop playing the greeting.
//       if "1" is entered, rewrite the prompt file and quit
//       if "2" is entered, play "Beep" and record the prompt.
//
entry Main is
  if AcceptCall() != null then stop; end if;
  PlayFile("GreetingIs");
  prompt = ReadStorageFile("MyGreeting.wav");
  loop
    if IsData(prompt) then Play(prompt); end if;
    input = ReadInput(10);
  exitif not IsString(input) or else input == "#";
    if input == "1" then
        if WriteStorageFile("MyGreeting.wav",prompt) then
          PlayFile("Goodbye"); stop;
        end if;
        PlayFile("Failure");
      elif input == "2" then
        PlayFile("Beep");
        prompt = Record(30);
      else
        PlayFile("InvalidEntry");
      end if;
  end loop;
  PlayFile("GoodBye");
end entry;
```

## Bridges and Mixers

StartBridge(*taskRef*)

This function sends a special *StartBridge* Event to the specified Task asking it to take over this Task peer media.

The *taskRef* value should be a task handle. It specifies the Task to send the request to.

This function returns a null-value if the specified Task successfully took over this Task peer media. Otherwise, the function returns an error code string.

The current Task should be in the `incoming`, `provisioned`, or `connected` mode.

The current Task is placed into the waiting state, and the target Task receives a special *StartBridge* Event.

IsStartBridgeEvent(*input*)

This function returns a true-value if the *input* value is a *StartBridge* Event. Otherwise the function returns a null-value.

RejectBridge(*input*,*errorCode*)

This function rejects the *StartBridge* request.

The *input* parameter value should be a *StartBridge* Event to reject.

The *errorCode* parameter value should be a string.

The StartBridge function in the Task that has sent this *StartBridge* Event exits the waiting state and it returns an error code string.

This function returns a null-value if the pending incoming call has been successfully rejected. Otherwise, the function returns an error code string.

AcceptBridge(*input*)

This function builds a Media Bridge with the Task that has sent it the *StartBridge* Event.

The *input* value should be the *StartBridge* Event to accept.

This function returns a null-value if the Media Bridge has been successfully established. Otherwise, the function returns an error code string.

The StartBridge function in the Task that has sent this *StartBridge* Event exits the waiting state. That function returns a null-value if the Media Bridge is established, otherwise, it returns an error code string. If that Task was in the `incoming` or `provisioned` mode, the call in accepted, and the Task switches to the `connected` mode.

When a Media Bridge is successfully established between a pair of Tasks, their peer media are connected to each other. Tasks Media Channels are disconnected from their peers and the Media Channel operations (`PlayFile`, `Record`, etc.) cannot be used. A Task cannot use the `StartBridge`, `AcceptBridge`, and `AttachMixer` functions while a Media Bridge is active.

BreakBridge()

This function removes the Media Bridge established with a successful completion of the `StartBridge` or `AcceptBridge` function.

The Task Media Channel is reconnected to the peer media.

A special *BreakBridge* Event is sent to the "bridged" Task.

This function returns a null-value if the existing media bridge has been broken. Otherwise, the function returns an error code string.

IsBreakBridgeEvent(*input*)

This function returns a true-value if the *input* value is a *BreakBridge* Event. Otherwise the function returns a null-value.

When a Task receives a *BreakBridge* Event, it does not have to use the BreakBridge() procedure, as the Media Bridge has been already removed.

If a Task disconnects its peer, or the Task peer disconnects itself, or a Task stops (normally, or because of an error), and there is an active Media Bridge, this Media Bridge is automatically removed.

```
//
// Sample: StartBridge()/AcceptBridge()/BreakBridge()
//   Accept an incoming call (stop if it's not possible).
//   Create a new Task to run the Caller code,
//     and send it an Event with the URI to dial.
//   Play the PleaseWait media file.
//   Wait for a StartBridge Event from the Caller Task.
//   Accept it and loop till the user disconnects.
//
//   The Caller code:
//   Receive a URI to call as an Event from the parent Task
//   Connect to the URI and play the YouGotACall media file
//   StartBridge with the parent, loop till the user disconnects
//
entry Caller forward;
procedure ControlBridge() forward;
```

```
entry Main is
   if AcceptCall() != null then stop; end if;

   callerTask = spawn Caller;
   if callerTask == null or else
      SendEvent(callerTask,"dial","sip:internal@partner.dom") != null then
         PlayFile("Failure");
         stop;
   end if;

   PlayFile("PleaseWait");
   input = ReadInput(30);
   if not IsStartBridgeEvent(input) or else
      AcceptBridge(input) != null then
         PlayFile("Failure");
         stop;
   end if;

   // we have established a bridge
   ControlBridge();
   PlayFile("GoodBye");
end entry;

//
// Caller Task code
//
entry Caller is
   // wait for a "dial" event from the main task
   input = ReadInput(30);
   if input == null or input.what != "dial" then stop; end if;

   mainTask = input.sender;

   // Calling the URI specified as the Event parameter
   // If connection failed, send an Event back to the
   //   main task and quit
   resultCode = StartCall(startEvent.parameter);
   if resultCode != null then
      void(SendEvent(mainTask,"result",resultCode));
         stop;
   end if;

   // wait for any Event other than provisional ones
   loop
      input = ReadInput(3600);
   exitif not IsCallProvisionEvent(input);
   end loop;

   // the parent has sent us "stop" - then we'll die immediately
   if IsDictionary(input) and then input.what == "stop" then stop; end if;

   if not IsCallCompletedEvent(input) or else input.parameter != null then
      void(SendEvent(mainTask,"result","generic error"));
         stop;
   end if;

   if StartBridge(mainTask) != null then
         PlayFile("Failure");
         stop;
   end if;

   // we have established a bridge
   ControlBridge();
   PlayFile("GoodBye");
end entry;

//
// Controlling the peer signaling:
// while the media is bridged:
//       exit if the peer hangs up, dials "#"
//   or if the bridge is removed
//
procedure ControlBridge() is
   loop
      input = ReadInput(3600);
   exitif IsBreakBridgeEvent(input) or else
          IsDisconnectEvent(input) or else input == "#";
   end loop;
   void(BreakBridge());
end procedure;
```

AttachMixer(*input*)

      The *input* value should be a StartBridge Event sent to this Task.

      This function attaches the peer media of the sender Task to this Task Media Channel.

      This function returns a null-value if the sender Task peer media is successfully attached to this Task Media Channel.

Otherwise, the function returns an error code string.

The StartBridge function in the sender Task exits the waiting state. That StartBridge function returns a null-value if the sender Task peer media is attached successfully. Otherwise, that function returns an error code string.

**Note:** If the `AttachMixer` function is used when the Task is in the *disconnected* mode, and the Task does not have a Media Channel, a new Media Channel is created for this Task.

`DetachMixer(`*`taskRef`*`)`

This function detaches the peer media of the specified Task from this Task Media Channel.

The *taskRef* value should be a task handle.

The function returns a null-value if the peer media of the specified Task was attached to this Task Media Channel, and if that peer media is successfully reconnected back to the specified Task.

The specified Task receives a BreakBridge Event.

The function returns an error code string if the other Task peer media cannot be detached.

The *taskRef* value can be a null-value. In this case, all other Tasks peer media are detached from this Task Media Channel.

`MixerAttached()`

This function returns an array of task handles for all Tasks that attached their peer media to this Task Media Channel.

If this Task Media Channel does not have any other Task peer media attached, this function returns a null-value.

`MuteMixer(`*`taskRef,flag`*`)`

This procedure tells the Task Media Channel if it should ignore input from the specified Task.

The *taskRef* value should be a Task handle. This Task should have its peer media attached to the current Task Media Channel.

The *taskRef* value can be a null-value, in this case the current Task peer media is controlled.

The *flag* value specifies the operation: if the flag is a true-value, the peer media is ignored, if it is a null-value, the Media Channel starts to process media from the peer.

If the *flag* value is the `"special"` string, the peer media is not distributed to other "normal" peers.

If the *flag* value is the `"hearsSpecial"` string, the peer mute mode is not modified, but the media from the "special" peers is distributed to it.

If the *flag* value is the `"hearsMute"` string, the peer mute mode is not modified, but the media from all peers (muted, special, or unmuted) is distributed to it.

If the *flag* value is the `"hearsNormal"` string, the peer mute mode is not modified, but the media from the "special" peers is not distributed to it.

When a Task has other Task peer media attached to its Media Channel, all media are placed into one *conversation space* (or a *conference*).

This Task cannot use the `StartBridge` or `AcceptBridge` functions.

**Note:** in certain cases the system may decide to convert an `AcceptBridge` function operation into an `AttachMixer` function operation. As a result, the `BreakBridge` operation can be used by a Task that has exactly one other peer media attached to its Media Channel.

If a Task disconnects its peer, or the Task peer disconnects itself, or a Task stops (normally, or because of an error), and there are other Task peer media attached to this Task Media Channel, the system automatically detaches all of them.

`StartBridgedCall(`*`destination,event`*`)`

This function works as the `StartCall` function, but the `event` value should be StartBridge Event sent to this Task by an *ingress* Task. The Task initiates a call using the media descriptor from the Event data (describing the ingress Task peer media).

Provisional responses are delivered as Events to the current Task, and they are sent to the ingress Task (see below).

If the call is successful, a Media Bridge between the Tasks is built,the current Task receives a final response Event, and the StartBridge operation in the ingress Task completes successfully.

If the call fails, the current Task receives a final response Event, but no Event is sent to the ingress Task. The current Task can either try a new StartBridgedCall operation, or it can use AcceptBridge/AttachMixer/RejectBridge operations to process the received StartBridge Event.

If the *ingress* Task quits, or if that Task had a pending incoming call and that call was canceled, the initiated outgoing call is canceled, and the current Task receives the *BreakBridge* event.

# Call Transfer

`TransferSupported()`
> This function returns a true-value if the peer supports Transfer operations, otherwise the function returns a null-value.

`IsCallTransferredEvent(`*input*`)`
> This function returns a true-value if the *input* value is a *CallTransferred* Event (this event can be sent when an external request has switched this Task to a different peer). Otherwise the function returns a null-value.

`TransferCall(`*destination*`)`
> This function attempts a "blind transfer" of the Task peer. The Task should be in the `connected` mode.
> The *destination* value should be a string containing the URI or an E-mail address to transfer the call to.
> If the call transfer fails, the function returns an error code string.
> If the call transfer succeeds, the function returns a null-value.
> When the operation is completed, the Task stays in the `connected` mode, unless the peer has disconnected explicitly (by sending the BYE request).

`StartTransfer(`*taskRef*`)`
> This function sends a special *StartTransfer* Event to the specified Task asking it to connect Task peers directly.
> The *taskRef* value should be a task handle. It specifies the Task to send the request to.
> This function returns a null-value if the specified Task successfully connected the peers. Otherwise, the function returns an error code string.
> The current Task should be in the `connected` mode.
> The current Task is placed into the waiting state, and the target Task receives a special *StartTransfer* Event.

`IsStartTransferEvent(`*input*`)`
> This function returns a true-value if the *input* value is a *StartTransfer* Event.
> Otherwise the function returns a null-value.

`RejectTransfer(`*input*`,`*errorCode*`)`
> This function rejects the *StartTransfer* request.
> The *input* parameter value should be a *StartTransfer* Event to reject.
> The *errorCode* parameter value should be a string.
> The StartTransfer function in the Task that has sent this *StartTransfer* Event exits the waiting state and it returns an error code string.

`AcceptTransfer(`*input*`)`
> This function connects the current Task peer with the peer of the Task that has sent it the *StartTransfer* Event.
> The *input* value should be the *StartTransfer* Event to accept.
> This function returns a null-value if the peers have been successfully connected. Otherwise, the function returns an error code string.
> The StartTransfer function in the Task that has sent this *StartTransfer* Event exits the waiting state. That function returns a null-value if the peers have been connected, otherwise, it returns an error code string.

If the peers have been connected, both Tasks stay in the `connected` mode, unless their peers explicitly send the disconnect Signals. The Tasks should either quit or they should use the `Disconnect` procedure to fully disconnect from their peers.

# Info Requests

Applications can receive and send INFO requests.

Certain INFO requests (such as DTMF event requests) are processed automatically and this section does not apply to them.

INFO request data is presented as a dictionary, with the following elements:

`Content-Type`
> This optional string element contains the request body Content-Type (such as `application`).
`Content-Subtype`
> This optional string element contains the request body Content-Type subtype.
`""` (empty key)
> This optional string or datablock or XML element contains the request body content.

SendCallInfo(*params*)
>  This function sends an INFO request to the Task peer. The Task should be in the `connected` mode.
>  The *params* value should be a dictionary containing the INFO request data.
>  If the operation fails, the function returns an error code string.
>  If the operation succeeds, the function returns a null-value.

When an INFO request is sent to a Task, the Task receives a special *CallInfo* Event. The Event `parameter` element contains a dictionary - the INFO request data.

IsCallInfoEvent(*input*)
>  This function returns a true-value if the *input* value is a *CallInfo* Event. Otherwise the function returns a null-value.

## Register Requests

Applications can send REGISTER requests to external SIP servers.

REGISTER request data is presented as a dictionary, with the following elements:

targetDomain
>  This string specifies the domain for registration.

Via
>  This optional element specifies the address of the remote server if it's different from the `targetDomain` value or can not be resolved from it.

fromAddress
>  This string specifies the Address Of Record (full account name) on the remote server.

Call-ID, CSeq
>  These string elements specify parameters for the registration dialog.

authUsername, authPassword
>  These string elements specify credentials for the registration.

`""` (empty key)
>  This string should specify the value for the Contact URI of the REGISTER request. That URI should be used by the remote SIP server as the target for the `INVITE` requests while processing calls coming in for the account on the remote system.

SendCallRegister(*params*)
>  This function sends an REGISTER request to a remote SIP server. The Task should be in the `disconnected` mode.
>  The *params* value should be a dictionary containing the REGISTER request data.
>  If the operation fails, the function returns an error code string.
>  If the operation succeeds, the function returns the TimeStamp when the requested registration expires.

Upon a successful REGISTER operation completion the function returns a TimeStamp value - the GMT time when the requested registration will expire. The task should re-schedule the next registration request for the time before that expiration time stamp, and for the next attempt use the same parameters, including `Call-ID`, and increment the `CSeq` value by 3.

## Service Requests

SendCallOptions(*params*)
>  This function sends an OPTION request to the Task peer (if the dialog has been established and the Task is in the `connected` mode), or to an arbitrary entity (if the Task is in the *disconnected* mode).
>  The *params* value is interpreted in the same way as the `StartCall` *destination* parameter value.
>  If the operation fails, the function returns an error code string.
>  If the operation succeeds, the function returns a null-value.

SignalOption(*optionName*)
SignalOption(*optionName*,*newValue*)
>  This function return the current value of a Task Signal option.
>  If the *newValue* value is specified and its value is not a null-value, then it is set as the new option value.
>  The *optionName* value should be a string and it specifies the option name. The following options are supported:

>  `"refer"`

this option specifies how the Task processes the [REFER requests](#) and [INVITE/replaces requests](#). The supported values are:

"self"
> outgoing signals are authenticated as coming from the current Account; this is the default value

"peer"
> outgoing signals are authenticated as coming from the Task peer (i.e. the sender of the REFER signal)

"disabled"
> REFER and INVITE/replaces request processing is prohibited.

"transferReport"
> this option specifies if call transfers are to be reported. The supported values are:

"NO"
> no event is sent when a Task is transferred to a different peer; this is the default value

true-value ("YES")
> the Task receives a special *CallTransferred* Event when it is transferred to a different peer.

"bridgeBreak"
> this option specifies how the Task reacts when a media bridge is broken by the other task. The supported values are:

"disconnect"
> the Task disconnects its current peer, and the Task enters the *disconnected* mode.

"keep"
> if the Task does not have a Media channel, it is created. Then the Task peer is [switched to that channel](#).

"default"
> if the Task does not have a Media channel, the Task disconnects its current peer. Otherwise, the Task peer is [switched to that channel](#). This is the default value.

"bridgedProvisionRelay"
> this option specifies how the provisional responses generated with the `StartBridgedCall` function are delivered to the peer. The supported values are:

true-value ("YES")
> provisional responses are relayed to this Task peer, using the ProvisionCall function with the `reliably` parameter set to a null-value; this is the default value

"reliably"
> provisional responses are relayed to this Task peer, using the ProvisionCall function with the `reliably` parameter set to a true-value

"NO"
> provisional responses are not relayed to this Task peer

"bridgedProvisionToTags"
> this option specifies how the provisional responses generated with the `StartBridgedCall` function are delivered to the peer. The supported values are:

"NO"
> all provisional responses contain the same To-tag, assigned to this Task; this is the default value.

true-value ("YES")
> the provisional responses keep their original To-tags.

`"bridgedProvisionQueue"`
> this option specifies the provisioning response FIFO queue size. This queue is used when provisioning responses from the other task are received faster that they can be delivered to this Task peer. The default queue length is 10.

`"callInfo"`
> this is a read-only option; if specified, the newValue must be a null-value.
> The function returns information about the system "call" object: time connected, proxies used, etc.

`SendCallNotify(`*`params`*`)`
> This function sends a NOTIFY request to an arbitrary entity (the Task must be in the *disconnected* mode).
> The *params* value must be a dictionary containing the following string elements:

`""` (empty string), `From`, `To`, `Call-ID`
> these elements are processed in the same way as the `StartCall` *destination* parameter elements.

`Event`
> the event name string

`@Event-Params`
> an optional dictionary with the parametes to put into the request `Event` field.

`Subscription-State`
> an optional string to put into the request `Subscription-State` field.

`@Content`
> an optional string or a datablock to send as the request body.

`Content-Type`, `Content-Subtype`
> optional strings for the request Content-Type field.

> If the operation fails, the function returns an error code string.
> If the operation succeeds, the function returns a null-value.

# Instant Messages and XMPP-type Requests

A Task can receive Instant messages and XMPP requests. They are delivered to the Task as Events.

`IsInstantMessageEvent(`*`input`*`)`
> This function returns a true-value if the *input* value is an Instant Message Event. Otherwise the function returns a null-value.
> The event parameter is a dictionary. It contains the same elements as the SendInstantMessage function *content* parameter, with the following additional elements:

`From`
> a string - the sender E-Mail address.

`peerName`
> an optional string - the sender "real name".

`fromInstance`
> an optional string - the sender "instance".

`To`
> a string - the original recipient E-Mail address.

`toInstance`
> an optional string - the recipient "instance".

an optional string - the authenticated identity of the sender.

redirector
an optional string - the authenticated identity of the message redirector (the Account that redirected this message using its [Automated Rules](#) or other means).

`IsXMPPIQEvent(`*`input`*`)`
This function returns a true-value if the *input* value is an XMPP IQ/presence Event. Otherwise the function returns a null-value.
The event parameter is a dictionary. It contains the same elements as the SendXMPPIQ function *content* parameter, with the following additional elements: `From`, `peerName`, `fromInstance`, `To`, `toInstance`, `authName`, `redirector`, which have the same meanings as in Instant Message Events.

# Supported Media Formats

The following audio file formats are supported:

`WAV` (data starts with the `RIFF` tag)
a file should contain a single `data` chunk with PCM 8-bit or 16-bit data, or GSM-encoded data.
`au` (data starts with the `.snd` tag)
a file should contain PCM 8-bit or 16-bit data, or 8-bit mu-Law data.
`gsm`
a file should contain a stream of 33-byte GSM 06.10-encoded frames.

# XIMSS Protocol

- **Protocol and Message Syntax**
- **Pre-Login Operations**
- **Login Operations**
- **Two-Factor Authentication**
- **Service Operations**
- **Mailbox Management**
- **Mailbox Operations**
- **Message Operations**
- **Account Management**
- **Secure Messaging (S/MIME)**
- **Contacts Operations**
- **Calendar Operations**
- **Signaling**
- **Instant Messaging**
- **Roster and Presence**
- **XMPP-style requests**
- **Preferences**
- **File Storage Operations**
- **Automated Rule Management**
- **Remote POP, Remote SIP Management**
- **Real-Time Application Management**
- **Directory**
- **Datasets**
- **Billing**
- **Application Helpers**
- **Banner Retrieval**
- **XML Data Formats**
  - `EMail`
  - `MIME`
  - `MIMEReport`
- **HTTP Access**
  - **Message Part Retrieval**
  - **Profile Retrieval**
  - **File Uploading**
  - **File Downloading**
  - **Data Export**
  - **Private Key and Certificate Export**
  - **Abnormal Termination**

The CommuniGate Pro Server implements the XML Interface to Messaging, Scheduling and Signaling (XIMSS) protocol.

The Interface is implemented with the XIMSS module supporting TCP/IP networks.

The protocol session examples use the s: marker to show the data sent by the Server, and the c: marker to show the data sent by a Client.

The "client-side" programming libraries for the XIMSS protocol can be found at the XIMSS Client web site.

# Protocol and Message Syntax

XML API clients should open clear-text or secure TCP connections to the CommuniGate Pro Server XML module. When a connection is established, both sides can send and receive *messages*.

Each message is a text string ending with a binary zero byte.

Each message should be formatted as an XML document.

A client asks the Server to take actions and/or to retrieve data by sending a *request* message. Each request message must contain the `id` attribute.

When the Server completes the requested operation, it sends back a *response* message:

response

> Attributes:
> `id`
>> the same as the `id` attribute of the request message.
>
> `errorText`
>> this optional attribute is present only if the operation has failed. It contains the error message text.
>
> `errorNum`
>> this optional attribute is present only if the operation has failed. It contains the numeric error code.

> Example:
> ```
> C:<noop id="A001"/>
> S:<response id="A001"/>
> C:<myCommand id="A002" myParam="user1@example.dom" />
> S:<response id="A002" errorText="unknown command" errorNum="500" />
> ```

A client can send the next request message without waiting for the current request response (pipelining).

The Server can send *data* messages to the client:

- when it processes a client request message;
  these messages are sent before the response message is sent;
  these messages include the same `id` attribute as the request message.
- when an asynchronous Server event (such as arrival of an Alert or an Instant Message) is delivered to the client session.
  these messages do not include any `id` attribute.

> Example:
> ```
> C:<noop id="A001"/>
> S:<alert gmtTime="20070502T083313" localTime="20070502T003313">Account is over quota</alert>
> S:<response id="A001"/>
> S:<alert gmtTime="20070502T083500" localTime="20070502T003500">Please logout, as we are shutting down.</alert>
> ```

**Note:** a Client must send some command to the Server at least once in 10 minutes, otherwise the Server closes the connection.

---

# Pre-Login Operations

After a connection with the Server is established, and before the client performs the `login` operation, the client can perform only the operations listed in this section.

When a connection is established, the Server takes the IP Address the client has connected to and selects the [Domain](#) this IP address is assigned to.

Operations in this section can explicitly specify an alternative target Domain: if it is found, the new target Domain is set and it is

used in the next operations.

listFeatures
This operation tells the Server to return a `features` message containing available communication and authentication options.

Attributes:
domain
optional target [Domain](#) name.

readStrings
This operation reads the vocabulary (words, tags, button names, etc.) stored in the current Domain [Skin](#). The Server sends a [strings](#) data message with the dictionary data.

Attributes:
skinName
the Skin name. If not specified, the unnamed Skin is used.

language
the vocabulary language. If the attribute is missing, the Account `Language` preference string is used.

timeModified
if this attribute is specified, it should contain a date and time value (GMT time). If this attribute is specified, the [strings](#) data message does not have any body element if the requested vocabulary modification time is not newer than this attribute value.

starttls
This operation tells the server to establish SSL/TLS security on this connection.

Attributes:
domain
optional target [Domain](#) name.

If the server returns a positive response, the client should start SSL/TLS negotiations immediately.

recoverPassword
This operation asks the server to E-mail the password of the specified Account to the E-mail address the Account user has specified.

Attributes:
domain
optional target [Domain](#) name.

userName
the Account name.

If the server returns a positive response, the Account password has been E-mailed.
**Note:** this operation introduces a delay before returning a response.

signup
This operation tells the Server to create a new Account.

Attributes:
domain
optional target [Domain](#) name.

userName
the new Account name.

`password`
> the new Account password.

`realName`
> a Real Name of the new Account owner (optional).

`recoverPassword`
> an E-mail address that can be used to recover a forgotten password (optional).

If the server returns a positive response, the Account has been created.
**Note:** this operation does not log user into a newly created Account.
**Note:** this operation introduces a delay before returning a response.

The Server sends the following data messages:

`features`
> This synchronous data message is sent when the Server processes the `listFeatures` operation.
>
> Body:
>> a set of XML elements:
>>
>> `domain`
>>> the target Domain name.
>>
>> `sasl`
>>> the text body of this element is the name of the SASL mechanism enabled for the target Domain.
>>
>> `starttls`
>>> if this element is present, the client can perform the `starttls` operation.
>>
>> `nonce`
>>> a valid "nonce" data. This "nonce" data can be used with SASL [HTTP Login](#) methods.
>>
>> `language`
>>> the text body of this element contains the default language selected for the target Domain. If this element is absent, the default (English) language is selected.
>>
>> `signup`
>>> if this element is present, the client can perform the `signup` operation with the target Domain.
>>
>> `connect`
>>> if this element is present, it specifies the connection method the client should use for the login and session requests. It is based on the Recommended XIMSS Method [Domain Setting](#) for the target Domain.
>>> The element has the following attributes:
>>>
>>> `protocol`
>>>> the protocol to use: `http` or `ximss`. The recommended protocol should be a secure one (such as https) if the `listFeatures` operation was issued over a secure protocol.
>>>
>>> `port`
>>>> the TCP port number to use. If not specified, the client should use the same port as it used to send this `listFeatures` operation.

# Login Operations

The client should perform the `login` operation to authenticate the user and to create a XIMSS session.

`login`

> Attributes:
> `domain`
>> optional target [Domain](#) name.
>
> `method`
>> this attribute specifies the [SASL](#) method to use. If this attribute is missing the *clear-text* (*password*) method is used.
>
> `authData`
>> If the *clear-text* method is used, this attribute contains the username.
>> For all other methods, this is a string with base64-encoded SASL protocol data.
>
> `password`
>> This attribute is used for the *clear-text* authentication method only, it contains the user password in the clear-text form.
>
> `version`
>> This attribute specifies the protocol version this Client is designed for (for example, `6.0`). If the version is not specified, the 5.4 version is assumed.

> Example:
> ```
> C:<login id="A001" authData="user1@example.dom" password="123rtu" version="6.1" />
> S:<session id="A001" urlID="12-skejlkieuoiuoi-dnciru" userName="user1@example.dom" realName="User J. Smith"/>
> S:<response id="A001"/>
> ```

The Server sends the following data messages:

`session`

> This message contains information about the newly created session. It is sent before the positive response for the `login` operation is sent.

> Attributes:
> `urlID`
>> the session ID string. This ID string can be used to access session data via [HTTP](#).
>
> `userName`
>> the authenticated Account full name (*accountName@domainName*).
>
> `realName`
>> the authenticated Account user Real Name (this attribute is absent if the Real Name is not specified in the Account Settings).
>
> `version`
>> the protocol version supported with the Server (for example, `6.1`).

> Examples:
> ```
> C:<login id="A001" authData="user1@example2.dom" password="rrr123" />
> S:<session id="A001" urlID="12-skejlkieuoiuoi-dnciru" userName="user1@example.dom" realName="User J. Smith"
> version="6.1.2" />
> S:<response id="A001"/>
> ```

When the specified SASL method involves sending a challenge to the client, it is sent as a `challenge` data message, with the `value` attribute containing the base64-encoded SASL protocol challenge data.
The client should respond by sending an `auth` request with the same `id` attribute as one used in the `login` request, and the `value` attribute containing the base64-encoded SASL protocol response data.

> Example (see RFC2195):
> ```
> C:<login id="A001" method="CRAM-MD5" />
> ```

```
S:<challenge value="PDE4OTYuNjk3MTcwOTUyQHBvc3RvZmZpY2UucmVzdG9uLm1jaS5uZXQ+" />
C:<auth id="A001" value="dGltIGI5MTNhNjAyYzdlZGE3YTQ5NWI0ZTZlNzMzNGGQzODkw" />
S:<session id="A001" urlID="12-skejlkieuoiuoi-dnciru" userName="user1@example.dom" realName="User J. Smith"/>
S:<response id="A001"/>
```

---

# Two-Factor Authentication

If Two-Factor Authentication is enabled and required to complete the <u>login</u> operation the <u>session</u> response contains the `x2auth` dictionary that describes the methods available for Two-Factor Authentication, with the key of each element being the method name and the value being the dictionary with method parameters:

Parameters:
address
>   obfuscated address where the one-time password will be sent to.
>   This parameter may be an array when one method supports multiple addresses.

icon
>   this optional attribute specifies the icon file name to represent this method in the user interface.

humanTag
>   this optional attribute specifies the name of the string element in the Skin to name this method in UI.

humanName
>   this optional attribute specifies the string to name this method in UI. It's used when `humanTag` element is not present.
>   When neither `humanTag` nor `humanName` are present, the raw method name can be used in UI.

Example:
```
C:<login id="A001" authData="user1@example2.dom" password="rrr123" x2auth="1" />
S:<session id="A001" urlID="14-skejlkieuoiuoi-dnciru" userName="user1@example.dom" realName="User J. Smith"
version="6.2c1">
S:<x2auth>
S: <subKey key="sms">
S:   <subKey key="address">+1555***2207</subKey>
S:   <subKey key="humanName">Send SMS to</subKey>
S: </subKey>
S: <subKey key="call">
S:   <subKey key="address">+1555***2207</subKey>
S:   <subKey key="humanTag">x2auth_voip_tag</subKey>
S:   <subKey key="icon">x2auth_voip.png</subKey>
S: </subKey>
S: <subKey key="email">
S:   <subKey key="address">d***6@gmail.com</subKey>
S:   <subKey key="humanTag">x2auth_email_tag</subKey>
S: </subKey>
S: <subKey key="push">
S:   <subKey key="address">
S:    <subValue>John's iPhone</subValue>
S:    <subValue>Samsung Galaxy S6</subValue>
S:   </subKey>
S:   <subKey key="humanTag">x2auth_push_tag</subKey>
S:   <subKey key="icon">x2auth_voip.png</subKey>
S: </subKey>
S:</x2Auth>
S:<session/>
S:<response id="A001"/>
```

The client should present the user interface to select a method and then use `x2AuthStart` command to start the second stage of authentication and `x2AuthComplete` to complete it.

x2AuthStart
>   This operation starts transmission of the one-time secret to the devices possessed by the user, using the specified method.
>   On a success the Server sends the <u>x2auth</u> data message.

>   Attributes:
>   index
>>       Optional index of element in the array of addresses available for the selected method.

>   Body:
>>       String with the selected method name.

`x2AuthComplete`

>       Body:
>>              Optional string with the one-time secret to confirm the second stage of authentication.

The server sends the following empty data message in response to the `x2AuthStart` command:

`x2auth`

>       Attributes:
>       `user`
>>              Set if the `x2AuthComplete` expects the one-time secret from user input to unlock the session. This is the default.
>
>       `background`
>>              Set when the one-time secret verification is expected to complete in background. In this case `x2AuthComplete` can be executed without one-time password to check for that completion.

---

# Service Operations

The following operations can be used to manage the client-server connection.

`noop`
>       This operation does not do anything and it always succeeds.

`bye`
>       This operation does not do anything and it always succeeds. After the response message is sent to the client, the Server closes the connection and destroys the current session.

`passwordModify`
>       This operation modifies the Account Password and/or the Password Recovery E-mail Address.
>
>       Attributes:
>       `oldPassword`
>>              the current Account password. The operation verifies this password before attempting any modification.
>
>       `newPassword`
>>              this optional attribute specifies the new password. The [password modification](#) operation must be allowed for this Account.
>
>       `recoveryEMail`
>>              this optional attribute specifies the new password recovery E-mail.

`setSessionOption`
>       This operation sets an option value for the current session.
>
>       Attributes:
>       `name`
>>              the option name
>
>       `value`
>>              the option value
>
>       The following session options are supported:
>
>       `reportMailboxChanges`

if the option value is `yes`, the Server sends `mailboxModified` data messages, for any other value, the Server stops sending these data messages.

`idleTimeout`
> the session idle time-out is set to the specified number of seconds or to a some larger value, if the specified value is unavailable.

`listUploaded`
> This operation retrieves information for one or all files from the ["uploaded file set"](). The Server sends uploaded file information in an `uploadedFile` data message (see below).

> Attributes:
> `uploadID`
>> if this optional parameter is specified, information is retrieved only for the file with this fileID.
>> If this parameter is not specified, information for all "uploaded file set" files is retrieved.

`clearUploaded`
> This operation removes one or all files from the ["uploaded file set"]().

> Attributes:
> `uploadID`
>> if this optional parameter is specified, only the file with this fileID is removed from the "uploaded file set".
>> If this parameter is not specified, all files are removed from the "uploaded file set".

`readStrings`
> This operation reads the vocabulary (words, tags, button names, etc.) stored on the Server. The Server sends a `strings` data message with the dictionary data (see below).

> Attributes:
> `language`
>> the vocabulary language. If the attribute is missing, the Account `Language` preference string is used.

> `timeModified`
>> if this attribute is specified, it should contain a date and time value (GMT time). If this attribute is specified, the `strings` data message does not have any body element if the requested vocabulary modification time is not newer than this attribute value.

`readConfigFile`
> This operation reads a configuration file stored on the Server. The Server sends a `configFile` data message with the configuration file data (see below).

> Attributes:
> `name`
>> the configuration name.

> `timeModified`
>> if this attribute is specified, it should contain a date and time value (GMT time). If this attribute is specified, the `configFile` data message does not have any body element if the requested configuration modification time is not newer than this attribute value.

`readTime`
> This operation reads the current time from the Server. The Server sends a `currentTime` message.

`readStatus`
> This operation reads the current Account status. The Server sends a `currentStatus` message.

**readSession**

This operation makes the Server to resend the <u>session</u> message (which is sent when a new session is created).

**listKnownValues**

This operation causes the Server to send a <u>knownValues</u> message. This message contains sets of "known values": known Time Zone names, character set names, etc.

**skinList**

This operation lists [Skins](#) in the current Domain.

Attributes:
filter
an optional attribute specifies a picture with the star (`*`) symbol used as wildcards. Only the Skin names matching this picture are reported.

The Server returns one <u>skinInfo</u> message for each Skin found.

**skinFileList**

This operation provides a list of the selected Skin file names.

Attributes:
skinName
the Skin name. If not specified, the unnamed Skin is used.

filter
an optional attribute specifies a picture with the star (`*`) symbol used as wildcards. Only the file names matching this picture are reported.

The Server returns one <u>skinFileInfo</u> message for each file found.

**skinFileRead**

This operation reads a file from the specified Skin in the current Domain.

Attributes:
skinName
the Skin name. If not specified, the unnamed Skin is used.

fileName
the name of file to read.

type
If this optional attribute exists and its value is `binary`, the file data is returned base64-encoded.

The Server returns a <u>skinFileData</u> message.

**cliExecute**

This operation executes a [Network CLI](#) command.

Attributes:
report
if this attribute is present, and its value is `xml`, the resulting object will be sent as its XML presentation.

Body:
the CLI command text

If the CLI command produces a result, the Server sends a <u>cliResult</u> message.

**retrieveXML**

This operation reads an XML document from a remote server. The current user must have the HTTP [Service](#) enabled.

Attributes:
url

the XML document URL (`http:` or `https:`).

`timeout`
> the operation time-out (in seconds).

If the document is successfully retrieved, the Server sends a <u>retrievedXML</u> message with the retrieved document content.

`httpCall`
> This operation sends an HTTP request to and receive an HTTP response from a remote server. The current user must have the `HTTP` <u>Service</u> enabled.
> This operation uses the specified URL, a composed request dictionary, and a maximum time-out of 30 seconds to call the <u>HTTP Client</u> module to execute an HTTP request.

> Attributes:
> `url`
> > the document URL (`http:` or `https:`).

> *other attributes*
> > other attributes are used to compose the request dictionary.

> Body:
> > If specified, it is used to compose the `body` element of the request dictionary (the HTTP request body). It should use one of the following formats:

> > a text body
> > > the text is used as the request body.

> > one `body` XML element
> > > the element text body is used as the request body.

> > one `base64` XML element
> > > the element text body is base64-decoded and the resulting datablock is used as the request body.

> > a set of `field` XML elements
> > > a dictionary with these elements is used as the request body (the HTTP POST request data).
> > > Each `field` element specifies a form field: the element `name` attribute specifies the form field name; the field value is specified with the `field` element body, which should be either a text or an <u>XML presentation</u> of a number, datablock, or a dictionary (see the <u>HTTP Client</u> module section for the details).

> > Besides these `body` data elements, the `httpCall` request body can contain:

> > one `supplFields` XML element
> > > the element body should contain one or more <u>subKey</u> elements; they form a dictionary specifying additional HTTP request fields.

> > one `urlParams` XML element
> > > the element body should contain one or more <u>subKey</u> elements - they form a dictionary specifying HTTP request URL parameters.

When the transaction completes, the Server sends a <u>retrievedURL</u> message with the transaction results.

`spellerList`
> This operation causes the Server to send <u>speller</u> messages, containing the names of installed spell checkers.

`spellerCheck`
> This operation checks the spelling of an arbitrary text. For each spelling error found, the Server sends a <u>spellerReport</u> message.

Attributes:

`speller`
>
> the spell checker name.

`mode`
>
> if this attribute exists and it has the `"glyphs"` value, offsets and lengths in the `spellerReport` messages define symbols (UTF8-encoded) rather than bytes.

Body:
>
> the text to check.

The Server can send the following service data messages at any time:

`alert`
>
> The authenticated account has received an [Alert](#). As soon as the Server sends the Alert data message to the Client, the Alert message is marked as "confirmed".
>
> Attributes:
>
> `gmtTime`
>> the time when the Alert was posted (GMT).
>
> `localTime`
>> the time when the Alert was posted (in the selected time zone).
>
> Body:
>> the Alert text (in the UTF-8 encoding).

`bye`
>
> The Server has decided to close a session (time-out, administrator action, etc.)
> The client is supposed to close its Server connection (if it has one), and to inform the user.

`newUploaded`
>
> A new file has been added to the session ["uploaded file set"](#).
>
> Attributes:
>
> `uploadID`
>> a string identifying the file in the "uploaded file set".

`mailboxModified`
>
> Some of the current Account mailboxes has been modified. These messages are sent only if the session option `reportMailboxChanges` value was set to `yes`.
>
> Attributes:
>
> `mailbox`
>> the modified mailbox name.

The Server can send the following data messages:

`uploadedFile`
>
> This synchronous data message is sent when the Server processes the `listUploaded` request.
>
> Attributes:
>
> `uploadID`
>> a string identifying a file in the "uploaded file set".

`fileName`
>> the uploaded file name (optional)

`size`
>> the uploaded file size

`type`
>> the uploaded file Content type

`subtype`
>> the uploaded file Content subtype (optional)

`charset`
>> the uploaded file Content charset (optional)

`strings`
> This synchronous data message is sent when the Server processes the <u>readStrings</u> request.

> Attributes:
> `skinName`
>> the Skin name.

> `language`
>> the selected language.

> `timeModified`
>> the vocabulary modification date and time (GMT time).

> Body:
>> a set of `string` and `strings` XML elements. Each element has the `name` attribute - the name (or "key") of the vocabulary element.
>> String-type vocabulary elements are presented using `string` elements. The `string` element body is the vocabulary element value.
>> Dictionary-type vocabulary elements are presented using `strings` elements. The `strings` element body is the set of `string` XML elements.
>> Array-type vocabulary elements are presented using `strings` elements. The `strings` element body is the set of `string` XML elements without the `name` attribute.

> Example:
> The Client reads the default (english) vocabulary.

```
C:<readStrings id="A001" />
S:<strings id="A001" language="" >
    <string name="AppendButton">Append</string>
    <strings name="AppendModes">
      <string name="simple">Simple Mode</string>
      <string name="advanced">Advanced Mode</string>
    </strings>
    <strings name="KnownFields">
      <string>From</string>
      <string>Subject</string>
    </strings>
  </strings>
S:<response id="A001"/>
```

`configFile`
> This synchronous data message is sent when the Server processes the <u>readConfigFile</u> request.

> Attributes:
> `skinName`
>> the Skin name.

> `name`
>> the configuration name.

> `timeModified`
>> the configuration modification date and time (GMT time).

Body:
>    the `config` XML element.


currentTime
>    These messages are sent when the Server processes the [readTime](#) request.

>    Attributes:
>    gmtTime
>>        the current Server time (GMT).

>    localTime
>>        the current Server time (in the selected time zone).


currentStatus
>    These messages are sent when the Server processes the [readStatus](#) request.

>    Body:
>>        a set of XML elements:

>>    messageStore
>>>        The information about the Account Message Storage.

>>>        Attributes:
>>>        used
>>>>            the currently used storage (in bytes).

>>>        limit
>>>>            the storage quota. This attribute is absent if the quota is set to Unlimited.


>>    PrevLogin
>>>        The information about the previous successful login.

>>>        Attributes:
>>>        gmtTime
>>>>            the login time (GMT).

>>>        localTime
>>>>            the login time (in the selected time zone).

>>>        ip
>>>>            the network address from which the user logged in.


>>    LastFailedLogin
>>>        The information about the last failed login.

>>>        Attributes:
>>>        gmtTime
>>>>            the time of the last failed login attempt (GMT).

>>>        localTime
>>>>            the time of the last failed login attempt (in the selected time zone).

>>>        ip
>>>>            the network address from which the failed attempt to log in was made.

RulesAllowed, SignalRulesAllowed, RPOPAllowed, PWDAllowed
> The effective Account settings data. These elements do not have attributes, and their text body is the setting value.

option
> zero, one, or several XML elements. The element body is a string, specifying an option enabled for the current Account:

> S/MIME
>> Secure Mail services are enabled.

> SMIMEActive
>> Secure Mail services are unlocked.

> SMIMEInactive
>> Secure Mail services are locked (but the Account does have a Private PKI key).

> WebCAL
>> Calendaring services are enabled.

> WebSite
>> Web access to File Storage is enabled.

> Signal
>> Signaling services are enabled.

> PBX
>> PBX services are enabled.

> HTTP
>> HTTP transaction Services are enabled.

signalBind
> The information about the session Real-Time parameters.

> Attributes:
> name
>> the "clientID" name assigned to this session.

timeout
> The information about the session time-out parameters.

> Attributes:
> limit
>> the session time limit, in seconds.

> idle
>> optional; the idle time-out, in seconds. If this attribute is not present, idle time-outs for this session are disabled.

knownValues
> These messages are sent when the Server processes the [listKnownValues](listKnownValues) request.

> Body:
> a set of XML elements:

> tzid

the element `name` attribute specifies a known Time Zone name;

charset
> the element `name` attribute specifies a known character set name;

mailRuleAllowed
> the element `name` attribute specifies a supported Allowed Mail Rule mode. Each mode defines which Mail Rule actions the user is allowed to modify; Elements are sorted, with the "most restrictive" mode listed first.

mailRuleCondition
> the element `name` attribute specifies a supported Mail Rule condition.

mailRuleAction
> the element `name` attribute specifies a supported Mail Rule action; the element `allowedSet` attribute specifies the enabling Allowed Mail Rule name. The user can modify a Rule containing this operation only if the user Account `RulesAllowed` setting value is the specified or a less restrictive Allowed Mail Rule mode.

signalRuleAllowed, signalRuleCondition, signalRuleAction
> the same elements as the `mailRuleAllowed`, `mailRuleCondition`, `mailRuleAction` elements, but for the Signal Rules.

skinInfo
> This synchronous data message is sent when the Server processes the [skinList](skinList) request.
>
> Attributes:
> skinName
>> the Skin name.

skinFileInfo
> This synchronous data message is sent when the Server processes the [skinFileList](skinFileList) request.
>
> Attributes:
> skinName
>> the Skin name.
>
> fileName
>> the file name.
>
> size
>> the file size in bytes.
>
> timeModified
>> an optional attribute with the file modification date and time (local time).

skinFileData
> This message is sent when the Server processes the [skinFileRead](skinFileRead) request.
>
> Attributes:
> fileName
>> the name of read file.
>
> timeModified
>> the file modification date and time (local time).
>
> Body:
>> Either a text with file data, or the `base64` element. The text body of this XML element is base64-encoded file data (base64 encoding allows you to retrieve binary data).

```
cliResult
```
These messages are sent when the Server processes the `cliExecute` request.

Body:
text or XML presentation of the CLI command output.

Example:
The Client reads some Account aliases.

```
C:<cliExecute id="A001">GETACCOUNTALIASES user@domain.com<cliExecute/>
S:<cliResult id="A001">(alias1,alias2)</strings>
S:<response id="A001"/>
C:<cliExecute id="A002" report="xml">GETACCOUNTALIASES user@domain.com<cliExecute/>
S:<cliResult id="A002"><subValue>alias1</subValue><subValue>alias2</subValue></strings>
S:<response id="A002"/>
```

```
retrievedXML
```
These messages are sent when the Server processes the `retrieveXML` request.

Attributes:
```
url
```
the document URL.

Body:
one XML element with the retrieved document.

```
retrievedURL
```
These messages are sent when the Server processes the `httpCall` request.

Attributes:
```
url
```
the document URL.

*other attributes*
the elements of the result dictionary returned with the HTTP Client module (excluding the `body` element).

Body:
the `body` element of the result dictionary:

the element is a string
the text body containing this string

the element is a datablock
a `base64` XML element containing base64-encoded datablock data

the element is an XML Object
the XML Object

```
speller
```
These messages are sent when the Server processes the `spellerList` request.

Attributes:
```
speller
```
the spell checker name; usually - the language name or the dialect name (such as `English-US` or `French-CA`).

spellerReport
These messages are sent when the Server processes the `spellerCheck` request and detects a spelling error.

Attributes:
position
the misspelled word position in the supplied text.

size
the misspelled word size in the supplied text (in bytes).

Body:
zero, one, or more `guess` XML elements. The text body of each element is a suggested replacement for the misspelled word.

**Note:** the supplied text is XML-decoded first, and the attributes specify the word position and size in the resulting decoded text (which must use UTF-8 character set).
**Note:** the position and size specify either bytes or symbols, depending on the request `mode` attribute value.

# Mailbox Management

A client can use the following set of operations to manipulate Mailboxes in the authenticated Account, as well as in other Accounts (by specifying the full Mailbox name as *~accountName@domainName/mailboxName*).

**Note:** all non-ASCII Mailbox names are specified using the UTF-8 charset (and not the IMAP UTF-7 encoding method).

mailboxCreate
This operation creates a new [Mailbox](#).

Attributes:
mailbox
this attribute specifies the new Mailbox name.

mailboxClass
this optional attribute specifies the Mailbox class.

mailboxRename
This operation renames a [Mailbox](#).

Attributes:
mailbox
this attribute specifies the existing Mailbox name.

newName
this attribute specifies the new Mailbox name.

children
if this optional attribute is present, all Mailbox "children" (nested Mailboxes) are renamed, too.

mailboxRemove
This operation removes a [Mailbox](#).

Attributes:
mailbox
an existing Mailbox name.

children
    if this optional attribute is present, all Mailbox "children" (nested Mailboxes) are removed, too.


mailboxList
    This operation makes the Server send a `mailbox` data message (see below) for each *visible* Mailbox (the Mailboxes for which the authenticated Account has the [lookup](#) access right).

    Attributes:
    filter
        this optional attribute specifies the filter string in the IMAP protocol format.

    mailboxClass
        if this optional attribute is specified, only the Mailboxes of the specified class are listed. Specify an empty string value to list only "mail"-class Mailboxes.

    pureFolder
        if this optional attribute exists, and its value is `yes`, then the result includes "pure" folders, if the value is `no`, "pure" folders are not included.
        See the `mailbox` data message description for the details.


mailboxSubList
    This operation makes the Server send a `mailboxSubscription` data message (see below) for each element in the Account *mailbox subscription* set. Note that Mailboxes in this set may or may not exist.

    Attributes:
    filter
        this optional attribute specifies the filter string in the IMAP protocol format.


mailboxSubUpdate
    This operation modifies Account *mailbox subscription* set.

    Body:
        a set of `mailboxSubscription` elements.

        Attributes:
        mailbox
            a Mailbox name in the UTF-8 character set.

        mode
            if this attribute value is `add` the `mailbox` name is added to the *Mailbox subscription* set (unless the set already contains this name).
            otherwise, the `mailbox` name is removed from the *mailbox subscription* set.


mailboxAliasList
    This operation makes the Server send a `mailboxAlias` data message (see below) for each [Mailbox Alias](#) in the current Account.

mailboxAliasUpdate
    This operation modifies the Account *Mailbox Aliases* set.

    Body:
        a set of `mailboxAlias` elements.

        Attributes:
        name
            the Mailbox Alias name in the UTF-8 character set.

mode
> if this attribute value is `add` the `name` Alias is added to the *Mailbox Aliases* set. If the set already contains an Alias with this name, it is replaced.
> otherwise, the `name` Alias is removed from the *Mailbox Aliases* set.

Body:
> a string: the target Mailbox name in the UTF-8 character set.

## mailboxRightsGet

This operation makes the Server send a `mailboxRight` data message (see below) with the user access rights for the specified Mailbox.

Attributes:
mailbox
> an existing Mailbox name.

## mailboxACLList

This operation makes the Server send a `mailboxACL` data message (see below) with the Mailbox Access Control List data.

Attributes:
mailbox
> an existing Mailbox name.

## mailboxACLUpdate

This operation modifies the Mailbox Access Control List for the specified Mailbox.

Attributes:
mailbox
> an existing Mailbox name.

Body:
> A set of `aclElem` XML elements:

> Attributes:
> pattern
> > the ACL element "name". It can be an Account name, a name with "+" or "-" prefix, etc. See the Mailbox Access Control Lists section for more details.

> mode
> > if this optional attribute value is `add`, the specified rights are added to the right set already specified for this ACL element. If the ACL element did not exist, it is created.
> > if this optional attribute value is `sub`, the specified rights are removed from the right set specified for this ACL element.
> > if this optional attribute value is `clear`, this ACL element is removed.
> > if this optional attribute has any other value, or if this attribute is absent, the specified rights replace the right set already specified for this ACL element. If the ACL element did not exist, it is created.

> Body:
> > a string; each symbol specifies a Mailbox Access Right.

The Server sends the following data messages:

`mailbox`
These synchronous data messages are sent when the Server processes the `mailboxList` request.

Attributes:
`mailbox`
the Mailbox name in the UTF-8 character set.

`UIDVALIDITY`, `MESSAGES`, `UIDNEXT`, `UNSEEN`, `OLDEST`, `CLASS`, `MEDIA`, `UNSEENMEDIA`
standard and extended **IMAP** Mailbox attributes

`SIZE`
Mailbox size (internal, same as INTERNALSIZE in IMAP).

`pureFolder`
this attribute exists and it has the `yes` value if the Mailbox is a pure folder - i.e. it cannot contain messages, but it can contain sub-Mailboxes.
**Note:** a Mailbox is seen as a pure folder if it can contain messages, but the Mailbox class does not match the class specified in the `mailboxList` request.

`isFolder`
this attribute exists and it has the `yes` value if the Mailbox is a mailbox and a folder - i.e. it can contain messages, and it can contain sub-Mailboxes.

`rights`
the effective access rights for the Mailbox. If this attribute is absent, access to the Mailbox is not restricted.

`mailboxSubscription`
These messages are sent when the Server processes the `mailboxSubList` request.

Attributes:
`mailbox`
the Mailbox name in the UTF-8 character set.

`mailboxAlias`
These messages are sent when the Server processes the `mailboxSubList` request.

Attributes:
`name`
the Mailbox Alias name in the UTF-8 character set.

Body:
a string; the target Mailbox name in the UTF-8 character set.

`mailboxRights`
This message is sent when the Server processes the `mailboxRightsGet` request.

Attributes:
`mailbox`
the Mailbox name.

Body:
a string; each symbol specifies an effective **Mailbox Access Right** granted to the current user.

mailboxACL
This message is sent when the Server processes the [mailboxACLList](#) request.

Attributes:
mailbox
the Mailbox name.

Body:
A set of `aclElem` XML elements, one element for each Mailbox Access Control List element:

Attributes:
pattern
the ACL element "name". It can be an Account name, a name with "+" or "-" prefix, etc. See the [Mailbox Access Control Lists](#) section for more details.

Body:
a string; each symbol specifies a [Mailbox Access Right](#) granted to or revoked from the "name".

Example:

- [A001](#): the Client asks the Server to create the `MyNotes` Notes-type Mailbox.
- [A002](#): the Client asks the Server to rename the `MyNotes` Mailbox into `SharedNotes`.
- [A003](#): the Client asks the Server to grant the Lookup and Select rights for the `SharedNotes` Mailbox to users `user1` and `user2`.
- [A004](#): the Client asks the Server to add the Delete and Insert rights for the `SharedNotes` Mailbox to user `user1` and to revoke the Select right from the user `user2`.
- [A005](#): the Client asks the Server to retrieve the ACL data for the `SharedNotes` Mailbox.

```
C:<mailboxCreate id="A001" mailbox="MyNotes" mailboxClass="IPF.StickyNote" />
S:<response id="A001"/>

C:<mailboxRename id="A002" mailbox="MyNotes" newName="SharedNotes" />
S:<response id="A002"/>

C:<mailboxACLUpdate id="A003" mailbox="SharedNotes">
    <aclElem pattern="user1">lr</aclElem>
    <aclElem pattern="user2">lr</aclElem>
  </mailboxACLUpdate>
S:<response id="A003"/>

C:<mailboxACLUpdate id="A004" mailbox="SharedNotes">
    <aclElem pattern="user1" mode="add">di</aclElem>
    <aclElem pattern="user2" mode="sub">r</aclElem>
  </mailboxACLUpdate>
S:<response id="A004"/>

C:<mailboxACLList id="A005" mailbox="SharedNotes"/>
S:<mailboxACL id="A005" mailbox="SharedNotes">
    <aclElem pattern="user1">lrdi</aclElem>
    <aclElem pattern="user2">l</aclElem>
  </mailboxACL>
S:<response id="A005"/>
```

# Mailbox Operations

A client can use the following operations to process a Mailbox in the authenticated Account, as well as in other Accounts (by specifying the full Mailbox name as *~accountName@domainName/mailboxName*).

folderOpen
This operation opens the specified Mailbox as a "Folder".
A Folder represents a Server Mailbox, with all messages being sorted and, optionally, filtered.
Each folder has a name, and one session cannot have two folders with the same name. On the other hand, the same session

can open the same Mailbox as two different folders (with different names). For example, an application may use one folder to show the Mailbox content sorted by the Date field, while maintaining a separate window where it shows the same Mailbox, but only the messages containing the `Business` tag in the Keywords field, with all these messages sorted by the From field.

When Mailbox messages are added, removed, or updated, the Server reports these updates to all clients that have opened that Mailbox as a Folder.

Each folder sends its update notifications independently, so the client does not need to know that two folders are presenting different views on the same Server Mailbox.

Attributes:
`folder`
> the name for the new Folder to be opened. A client can use an arbitrary string as a Folder name.

`mailbox`
> the Mailbox name. If this attribute is not specified, the folder name is used.

`mailboxClass`
> an optional attribute. If the specified Mailbox does not exist and this optional attribute is specified, the specified Mailbox is created.
> If this attribute value is a non-empty string, that value is assigned as the [Mailbox Class](#) to the newly created Mailbox.

`sortField`
> the name of a message header field to use for Mailbox sorting.

`sortOrder`
> if this optional attribute is specified and it has the `desc` value, the sorting order is reversed.

`filter`
> if this optional attribute is specified, only the Mailbox messages matching this attribute value are included into the Folder.

`filterField`
> if this optional attribute is specified, its value specifies the message header field to compare with the filter attribute. Only the messages containing the specified field and with the field value matching the filter attribute value are included into the folder.
> If this attribute value is `FLAGS`, the `filter` attribute value should contain a comma-separated list of [message flags names and/or negative names](#). Only the messages with the specified flags set and the specified negative flags not set are included into the Server view.
> For example, the `filter="Media,Unseen"` attribute will tell the Server to build a view using only the messages with the `Media` flags set and the `Seen` flag not set.
> If this attribute value is `body`, messages containing the filter attribute value in any message body part are included into the server view.
> If this attribute is missing, messages containing the filter attribute value in message body part, or in any message header field are included into the server view.

`hideDeleted`
> If this optional attribute is present, the "hide deleted" folder mode is switched on if the attribute value is `yes`. If this optional attribute is absent, the "hide deleted" folder mode is switched on if the current effective value of the DeleteMode Account Preference is not `Mark`.
> When the "hide deleted" mode is on, a mailbox message is removed from the folder list as soon as the "Deleted" flag is assigned to it. If the "Deleted" flag is removed later, the message does not re-appear in the folder, but it will become visible again after the folder is re-opened.

`UIDValidity`, `UIDMin`
> If these optional numeric attributes are present, and the Folder Mailbox UIDValidity value is equal to the UIDValidity request attribute value, then only the messages with UID not smaller than the UIDMin value are included into the Folder.

Body:
> the request body should contain one or more `<field>` elements.
> Each element body should contain a name of a header field to be retrieved for each message.

These fields are called *viewer fields*.

The following `sortField` and *viewer fields* can be specified:

- `From`, `To`, `Cc`, `Bcc`, `Reply-To`, `Sender`, `Return-Path`, or other *Email-field* name. If a message header contains the specified *Email-field*, it is parsed. If the field contains a "comment" (i.e. a "real name"), it is used. Otherwise, the parsed E-mail address is used.
- `Date`, `Resent-Date`. The field date value is converted to GMT. When elements with these field values are sent within a `folderReport` message, they contain the `localTime` attribute specifying the field value in the Time Zone selected for the current user.
- any other E-mail header field name (`Subject`, `X-Mailer`, etc). The field MIME-decoded value is used.
- `E-From`, `E-To`, or other `E-`*Email-field* name. If a message header contains the specified *Email-field*, it is parsed, and the parsed E-mail address is used.
- `Pty`. The `X-Priority` field value is converted to the strings `High`, `Low`, or to an empty string.
- `UID`. The message metadata - its UID (unique ID) in the Mailbox.
- `ORIGUID`. The message metadata - its original UID in the Mailbox. Unless the message is an updated version of some older (and now deleted) message, the ORIGUID value is the same as the UID value.
- `SIZE`. The message metadata - its "raw" size in the Mailbox.
- `INTERNALDATE`. The message metadata - its "timestamp". It is formatted in the same way as the `Date` element.
- `FLAGS` the message metadata attribute values are used.
  See the [Mailboxes](#) section for more information on the Mailbox message flags.

A session can use several open Folders at the same time.

A client should maintain an *internal view* of the Folder - an array or a table with one element for each Folder message. A client should keep that view synchronized with the Server view (implemented with that Folder).

When a Folder is opened, the Server sends a `folderReport` data message containing the total number of messages in the Folder. The client uses this number to create the initial internal view table, filling it with empty elements.

To display the Mailbox content on the screen, the client checks which elements of its internal view table it should use. If some of those elements are empty, the client should ask the Server to send it the information about the Folder messages specified by their index values (positions) in the sorted view (in the Folder).

Some Mailbox operations use a *message set* to specify the Mailbox messages to apply the operation to. Messages can be specified either by their UIDs, or by their index numbers (their positions in the folder view).

To specify messages by their UIDs, use one or more `UID` elements.
Each `UID` element can include one message, in this case the message UID is specified as the element body: `<UID>12377</UID>`.
An `UID` element can include a range of message UIDs specified as the `from` and `till` attributes: `<UID from="12377" till="12455" />`. The operation is applied to all Mailbox messages with UIDs not smaller than the `from` attribute value and not larger than the `till` attribute value. Please remember that Folder messages are not sorted by their UIDs, unless the `sortField="UID"` attribute was used in the [folderOpen](#) operation.

To specify messages by index, use one or more `index` elements.
Each `index` element can include one message, in this case the message index is specified as the element body: `<index>14</index>`.
An `index` element can include an index range specified as the `from` and `till` attributes: `<index from="12" till="10000" />`. The operation is applied to all Folder messages with position (index) not smaller than the `from` attribute value and not larger than the `till` attribute value.

A *message set* can include one or more `UID` elements, or it can include one or more `index` elements, but it cannot include both `UID` and `index` elements.

`folderBrowse`
This operation makes the Server send data messages for the specified message set in an open Folder.

Attributes:

`folder`
>     the Folder name.

Body:
>     a *message set* (see above).

The Server sends a `folderReport` data message for each message with the index or UID in the specified set. The data message attributes specify the message index (position) in the Folder and the message UID.
The `folderReport` data message body contains the `viewer fields` values.

The "on demand" client-server synchronization model is used. When a Folder message is modified, added, or deleted, the client gets an asynchronous `folderReport` data message with the `mode` attribute value set to `notify` (a "notify-mode" message).

The newly added messages do not become visible in the logical Folder and the deleted messages are not removed from the logical Folder. Requests to retrieve deleted message data return no data items or empty data items.

When the client application is ready to update its "internal view", it should use the `folderSync` operation:

`folderSync`
>     This operation tells the Server to send all pending Folder modifications to the client.

Attributes:

`folder`
>     the Folder name.

`limit`
>     optional; the maximum number of data messages to return.

The Server sends `folderReport` data messages with the `mode` attribute set to the `removed`, `added`, `updated`, or `attrsUpdated` value.

After the Server sends a "notify-mode" message to the Client, the Server may choose not to send further "notify-mode" messages for this Folder until the client performs the `folderSync` operation on this Folder.

If the limit attribute is specified and the specified number of data message has been sent, the Server finishes the the `folderSync` operation, but expects the client to perform an additional `folderSync` operation on this Folder before it sends further "notify-mode" messages for this Folder.

The client can send requests to the Server asking for certain update operations (such as message deletion), but it should update its internal view only when the Server sends it a `folderReport` data message informing the client about actual changes in the Folder.

**Note:** unlike UIDs, Folder message index can change any time when the `folderSync` operation is performed. Before you start any operation using an index-based message set, make sure that you have correctly updated the internal view with all `folderReport` data messages generated with the previously sent `folderSync` operation.

**Note:** when a message has been removed from or added to a Mailbox, the Server only sends `folderReport` data messages with the `mode="notify"` attribute. The Server-side "view" of the modified Folder (the logical Folder) is not updated, and it is safe to use an index-based message set to start a Folder operation.

`messageCopy`
>     This operation copies messages from an open Folder to a target Mailbox.
>     Note: the target is specified as a Mailbox, not as a Folder.

Attributes:

`folder`
>     the source Folder name.

`targetMailbox`
>    the target Mailbox name.

`mailboxClass`
>    If the target Mailbox does not exist and this optional attribute is specified, the target Mailbox is created. If this
>    attribute value is a non-empty string, the value is assigned as the [Mailbox Class](#) to the newly created Mailbox.

`encrypt`
>    if this optional attribute exists, and its value is `yes`, the messages are copied encrypted, using the current user
>    S/MIME certificate.

`decrypt`
>    if this optional attribute exists, and its value is `yes`, and the `encrypt` attribute is not specified, the messages are copied
>    decrypted. The Account Private Key should be unlocked, otherwise an error is returned.

`report`
>    If this optional attribute is specified, it should have the `uid` value.
>    For each message copied to the target mailbox, the Server sends the synchronous `messageAdded` message containing
>    the UID of the copied message.

Body:
>    a *message set* (see above).

`messageMove`
>    This operation is the same as the `messageCopy` operation, but if the messages have been copied successfully, the operation
>    deletes the original messages.

>    This operation can also be used to implement the "encrypt/decrypt message" functionality: the selected messages should be
>    moved to the same Mailbox using the `encrypt` or `decrypt` operation attribute.

`messageMark`
>    This operation modifies [Mailbox message flags](#) in an open Folder.

>    Attributes:
>    `folder`
>>    the Folder name.

>    `flags`
>>    this attribute specifies the Mailbox message flags to add or delete. Several operations can be specified, separated with
>>    the comma symbol. See the [Mailbox](#) section for more details.

Body:
>    a *message set* (see above).

`messageRemove`
>    This operation removes messages from an open Folder.

>    Attributes:
>    `folder`
>>    the Folder name.

>    `mode`
>>    an optional attribute, specifying the delete operation method. If specified, it should have one of the following values:

>>    - `Immediately` - the messages are physically removed from the Folder Mailbox.
>>    - `Move To Trash` - the messages are moved to the Trash Mailbox (its name is specified with the `TrashBox`
>>      Account preference). If the specified Folder is the Trash Mailbox itself, the messages are physically removed.
>>      `Mark`

- the messages are marked with the *Deleted* flag.

If this attribute is not specified, the method specified with the DeleteMode Account Preference is used.

Body:
> a *message set* (see above).

`folderExpunge`
> This operation removes all messages marked with the `Deleted` flag from the Folder Mailbox.
> Note: it removes ALL marked Mailbox messages, not only the messages visible in this Folder.
>
> Attributes:
> `folder`
>> the Folder name.

`folderClose`
> This operation closes an open Folder.
>
> Attributes:
> `folder`
>> the Folder name.

`folderReopen`
> This operation re-builds an open Folder by re-scanning its Mailbox using different sorting and filtering parameters.
>
> Attributes:
> `folder`
>> the Folder name.
>
> `sortField`, `sortOrder`, `filter`, `filterField`, `hideDeleted`, `UIDValidity`, `UIDMin`
>> these attributes have the same meaning as the same [folderOpen](#) operation attributes.
>
> Body:
>> if the request body contains at least one `<field>` element, these elements specify the new *viewer fields* set. If no `<field>` element is specified, the *viewer fields* set is not modified.

When a client uses this command, the Server sends the `folderReport` message with the `mode` attribute set to `init`, notifying the client about the new Folder size.
The client should clear its internal Folder view, and re-populate it again.

**Note:** if a Client receives a `folderReport` message with the `mode` attribute set to `notify` when the Client has already sent the `folderReopen` command, the Client should use the `folderSync` command after it receives response for the `folderReopen` command.

`emptyTrash`
> This operation physically removes all messages from the Trash Mailbox (if it has been specified).

`emptyMailbox`
> This operation physically removes all messages from the specified mailbox.
>
> Attributes:
> `mailbox`
>> the mailbox name.
>
> `duration`

time limit (in "time delta" format) between the current time and the internal date of the newest message to keep in the mailbox. Specify `0` to delete all messages.

The Server sends the following data messages:

`folderReport`
> These messages are sent synchronously when a Folder is being opened or re-opened, and when the client sends a `folderBrowse` or a `folderSync` request.
> These messages are sent asynchronously when a Folder status changes (messages are added, removed, or updated), in this case its `mode` attribute is set to `notify`.

> Attributes:
> `folder`
>> the Folder name.

> `mode`
>> this optional attribute specifies the type of notification.

>> - If the attribute value is `init` then the `messages` attribute specifies the total number of messages in the Folder, and the `unseen` attribute specifies the total number of unseen messages (messages without the Seen flag) in the Folder.
>>   This is a synchronous data message sent when a Folder is being opened or re-opened.
>> - If the attribute value is `notify` some Mailbox messages have been added, modified, or deleted.
>>   This message is sent as an asynchronous data message.
>>   The client is expected to send the `folderSync` request for this Folder.
>> - If the attribute value is `removed` then the `index` and `UID` attributes specify the message removed from the Folder.
>>   This data message is sent only in response to the `folderSync` request.
>>   The client should immediately update its internal view: for all messages with the index value larger than the specified `index` attribute value, the index value is decreased by 1.
>> - If the attribute value is `added` then the `index` and `UID` attributes specify the message added to the Folder.
>>   This data message is sent only in response to the `folderSync` request.
>>   The client should immediately update its internal view: for all messages with the index value equal or larger than the specified `index` attribute value, the index value is increased by 1.
>> - If the attribute value is `updated`, or `attrsUpdated`, or the attribute is missing, then the `index` and `UID` attributes specify a Mailbox message and the body contains the message *viewer field* values.

>> This attribute is present and it is set to `error` when the `folderBrowse` request fails to retrieve the message data (for example, when a message has been already deleted, but it has not been removed from the view, because the `folderSync` request has not been issued yet.

> `index`
>> the message index in this Folder.
>> This attribute is not present when the `mode` attribute value is `init`.

> `UID`
>> the message UID (unique ID).
>> This attribute is not present when the `mode` attribute value is `removed`, `updated`, `attrsUpdated`, or `added`.

> `messages`
>> the total number of messages in the Folder.
>> This attribute is present when the `mode` attribute value is `init`, `removed`, or `added`.

> `unseen`
>> the total number of unseen messages in the Folder.
>> This attribute is present when the `mode` attribute value is `init`, `removed`, `updated`, `attrsUpdated`, or `added`.

> `rights`
>> the effective access rights for the Folder.
>> This attribute is present when the `mode` attribute value is `init`. If this attribute is absent, access to the Folder is not

restricted.

`UIDValidity`, `UIDNext`
the Folder Mailbox UIDValidity and UIDNext values.
These attributes are present when the `mode` attribute value is `init`.

Body:
The body is present when the `mode` attribute value is absent (`folderBrowse` response message), or when the `mode` attribute value is `updated`, `attrsUpdated`, or `added` (`folderSync` response message).
The body contains a set of XML elements with *viewer field* values.

Example 1:

- A001: the Client asks the Server to open the INBOX Mailbox as Folder "INBOX" and to sort it by the INTERNALDATE 'field' (this is not an actual message field, but a message metadata element - it specifies the time when the message was added into the Mailbox). The Client informs the Server that it needs to retrieve the FLAGS, From, and Subject fields of Mailbox messages.
- the Server opens the INBOX Mailbox and sorts it; the Server informs the Client that the Folder has 234 messages.
- A002: the Client asks the Server to send it data from the first 4 Folder messages, and the Server sends that information to the Client.
- while the Server was processing this request, one message was added to the Mailbox.
- after the Server has sent the response message, the Folder messages number 2 and 35 were deleted.
- A003: the Client asks the Server to send it data from the first 4 messages in the sorted view (again).
- A004: the Client asks the Server to send it all Folder modifications.
- The Server informs the Client about the message number 35 being removed. The Client should remove the element number 35 from its internal view table and update the information on its screen if necessary. The total number of Folder messages has changed to 233.
- The Server informs the Client about the message number 2 being removed. The Client should remove the element number 2 from its internal view table. The message number 3 becomes the message number 2, the message number 4 becomes the message number 3, etc.
- The Server adds a newly added message to the Folder and informs the Client that it has inserted the message as the message number 1. The Client should update its internal view table by inserting a new element as the element number 1. The element number 1 becomes the element number 2, the element number 2 becomes the element number 3, etc.
- A005: the Client asks the Server to send it data from the first 4 messages in the Folder (again).

```
C:<folderOpen id="A001" folder="INBOX" mailbox="INBOX" sortField="INTERNALDATE" sortOrder="asc">
  <field>FLAGS</field><field>From</field><field>Subject</field>
  </folderOpen>
S:<folderReport id="A001" folder="INBOX" mode="init" messages="234" unseen="12" />
S:<response id="A001"/>

C:<folderBrowse id="A002" folder="INBOX"><index from="0" till="3" /></folderBrowse>
S:<folderReport id="A002" folder="INBOX" index="0" UID="123">
   <FLAGS>Seen,Deleted</FLAGS><From>John H. Smith</From>
   <Subject>Hello - just a test</Subject>
  </folderReport>
S:<folderReport id="A002" folder="INBOX" index="1" UID="543">
   <FLAGS>Seen,Answered</FLAGS><From>Jim Spammer</From>
   <Subject>This is the best offer!</Subject>
  </folderReport>
S:<folderReport id="A002" folder="INBOX" index="2" UID="343">
   <FLAGS>Seen,Media</FLAGS><From>user@example.com</From>
   <Subject>Meeting reminder</Subject>
  </folderReport>
S:<folderReport folder="INBOX" mode="notify" />
S:<folderReport id="A002" folder="INBOX" index="3" UID="512">
   <FLAGS>Seen,Flagged</FLAGS><From>Admin@hq.example.com</From>
   <Subject>Shutdown @ 4:45AM</Subject>
  </folderReport>
S:<response id="A002" />

S:<folderReport folder="INBOX" mode="notify" />

C:<folderBrowse id="A003" folder="INBOX"><index from="0" till="3" /></folderBrowse>
S:<folderReport id="A003" folder="INBOX" index="0" UID="123">
   <FLAGS>Seen,Deleted</FLAGS><From>John H. Smith</From>
   <Subject>Hello - just a test</Subject>
  </folderReport>
S:<folderReport id="A003" folder="INBOX" index="1" UID="543">
   <FLAGS>Seen,Answered</FLAGS><From>Jim Spammer</From>
   <Subject>This is the best offer!</Subject>
  </folderReport>
```

```
S:<folderReport id="A003" folder="INBOX" index="2" UID="343">
    <FLAGS>Seen,Media</FLAGS><From></From>
    <Subject></Subject>
  </folderReport>
S:<folderReport id="A003" folder="INBOX" index="3" UID="512">
    <FLAGS>Seen,Flagged</FLAGS><From>Admin@hq.example.com</From>
    <Subject>Shutdown @ 4:45AM</Subject>
  </folderReport>
S:<response id="A003" />

C:<folderSync id="A004" folder="INBOX" />
S:<folderReport id="A004" folder="INBOX" index="35" UID="117" mode="deleted" messages="233" unseen="11" />
S:<folderReport id="A004" folder="INBOX" index="2" UID="343" mode="deleted" messages="232" unseen="11" />
S:<folderReport id="A004" folder="INBOX" index="1" UID="976" mode="added" messages="233" unseen="12" >
    <FLAGS>Recent</FLAGS><From>CGatePro Discussions</From>
    <Subject>[CGP] Re: Session Timer?</Subject>
  </folderReport>
S:<response id="A004" />

C:<folderBrowse id="A005" folder="INBOX"><index from="0" till="3" /></folderBrowse>
S:<folderReport id="A005" folder="INBOX" index="0" UID="123">
    <FLAGS>Seen,Deleted</FLAGS><From>John H. Smith</From>
    <Subject>Hello - just a test</Subject>
  </folderReport>
S:<folderReport id="A005" folder="INBOX" index="1" UID="976">
    <FLAGS>Recent</FLAGS><From>CGatePro Discussions</From>
    <Subject>[CGP] Re: Session Timer?</Subject>
  </folderReport>
S:<folderReport id="A005" folder="INBOX" index="2" UID="543">
    <FLAGS>Seen,Answered</FLAGS><From>Jim Spammer</From>
    <Subject>This is the best offer!</Subject>
  </folderReport>
S:<folderReport id="A005" folder="INBOX" index="3" UID="512">
    <FLAGS>Seen,Flagged</FLAGS><From>Admin@hq.example.com</From>
    <Subject>Shutdown @ 4:45AM</Subject>
  </folderReport>
S:<response id="A005" />
```

Example 2:

- A010: the Client asks the Server to copy 3 messages from the already opened INBOX Mailbox to the Archive Mailbox
- The Archive Mailbox appears to be opened, and the Server sends a Mailbox notification message to the Client

```
C:<messageCopy id="A010" folder="INBOX" targetMailbox="Archive">
    <UID>512</UID><UID>123</UID><UID>976</UID>
  </messageCopy>
S:<folderReport folder="Archive" mode="notify" />
S:<response id="A010"/>
```

Example 3:

- A020: the Client asks the Server to mark 3 messages (UIDs 512, 123, and 976) with the Deleted flag.
- The Server sets these flags, and it sends a Mailbox notification message to the Client.
- A021: the Client asks the Server to send it all Mailbox modifications.
- The Server sends the updated information for the messages with UID 512 and 976. The message with the UID 123 already had the Deleted flag set, so this request has not modified that message and the Server does not send information for this message.
- A022: the Client asks the Server to expunge the Mailbox.
- The Server deletes the messages with UIDs 512, 123, and 976, as well as message with UIDs 446 and 756 which also happened to have the Deleted flag set. The Server sends a Mailbox notification message to the Client.
- A023: the Client asks the Server to send it all Mailbox modifications.
- The Server sends data messages informing the client that it has removed the messages with UIDs 512, 123, 976, 446, and 756 from its sorted Mailbox view.
- A024: the Client closes the INBOX Mailbox.

```
C:<messageMark id="A020" folder="INBOX" flags="Deleted">
    <UID>512</UID><UID>123</UID><UID>976</UID>
  </messageMark>
S:<folderReport folder="INBOX" mode="notify" />
S:<response id="A020"/>

C:<folderSync id="A021" folder="INBOX" />
S:<folderReport id="A021" folder="INBOX" index="1" UID="976" mode="updated" unseen="12" >
    <FLAGS>Recent,Deleted</FLAGS><From>CGatePro Discussions</From>
    <Subject>[CGP] Re: Session Timer?</Subject>
  </folderReport>
S:<folderReport id="A021" folder="INBOX" index="3" UID="512" mode="updated" unseen="12" >
    <FLAGS>Seen,Deleted,Flagged</FLAGS><From>Admin@hq.example.com</From>
    <Subject>Shutdown @ 4:45AM</Subject>
  </folderReport>
S:<response id="A021"/>

C:<folderExpunge id="A022" folder="INBOX" />
S:<folderReport folder="INBOX" mode="notify" />
S:<response id="A022"/>

C:<folderSync id="A023" folder="INBOX" />
```

```
S:<folderReport id="A023" folder="INBOX" index="0" UID="123" mode="deleted" messages="232" unseen="12" />
S:<folderReport id="A023" folder="INBOX" index="0" UID="976" mode="deleted" messages="231" unseen="11" />
S:<folderReport id="A023" folder="INBOX" index="29" UID="446" mode="deleted" messages="230" unseen="11" />
S:<folderReport id="A023" folder="INBOX" index="123" UID="756" mode="deleted" messages="229" unseen="11" />
S:<folderReport id="A023" folder="INBOX" index="1" UID="512" mode="deleted" messages="228" unseen="11" />
S:<response id="A023"/>

C:<closeMailbox id="A024" folder="INBOX" />
S:<response id="A024"/>
```

# Message Operations

A client can use the following set of operations to retrieve Messages from Mailboxes.

folderRead
>    This operation tells the Server to retrieve a Message or its part. It is sent to the client as a folderMessage data message.

>    Attributes:
>    folder
>>        the Folder name.

>    UID
>>        the message UID.

>    partID
>>        this optional attribute specifies the message part to read. If it is not specified, the entire message is read.

>    mode
>>        if this optional attribute is set to replyFrom or replyAll, the partID attribute should either be absent or it should specify an EMail subpart element. The folderMessage response message body is an EMail element with a pre-composed reply E-mail.

>    totalSizeLimit
>>        the maximum total size of all message parts returned.

>    x509
>>        if this optional attribute is set to base64, the x509 subelements in the retrieved messages (digital signatures) will contain the base64-encoded certificate data.

messageAppend
>    This operation tells the Server to compose a Message and append it to an open folder or to an arbitrary Mailbox.

>    Attributes:
>    folder
>>        the target Folder name. If specified, the targetMailbox attribute is ignored.

>    targetMailbox
>>        the target Mailbox name. It must be specified if the folder attribute is absent.

>    mailboxClass
>>        this optional attribute can be specified if the targetMailbox attribute is specified.
>>        If the target Mailbox does not exist and this optional attribute is specified, the target Mailbox is created. If this attribute value is a non-empty string, the value is assigned as the Mailbox Class to the newly created Mailbox.

>    flags
>>        this optional attribute specifies the message flags the newly created message will have in the Mailbox. Several flags can be specified, separated with the comma symbol. See the Mailbox section for more details.

>    internalDate
>>        this optional attribute specifies the "internal timestamp" for the newly created message. If this parameter is not specified, the current time is used.

>    replacesUID
>>        this optional attribute can be specified if the folder attribute is specified.
>>        If the message has been successfully composed and appended to the Folder, the message with the replacesUID UID is removed from the Folder Mailbox.

replaceMode
> this optional attribute can be specified if the `replacesUID` attribute is specified.
> if the attribute value is `checkOld`, the the operation fails if the Folder Mailbox does not contain a message with `replacesUID` real UID. This check and the append/delete operations are executed atomically.
> if the attribute value is `copyOrigUID`, the operation fails if the Folder Mailbox does not contain a message with `replacesUID` real UID. If the message does exist, then its ORIGUID attribute is copied to the newly created message. This check, the append/delete operations, and the attribute assignment are executed atomically.
> if the attribute value is `origUID`, replaceUID attribute specified the ORIGUID attribute of the message to be replaced. The operation fails if the Folder Mailbox does not contain a message with `replacesUID` ORIGUID. If the message does exist, then its ORIGUID attribute is copied to the newly created message. This check, the append/delete operations, and the attribute assignment are executed atomically.

report
> If this optional attribute is specified, it should have the `uid` value.
> If the message has been successfully composed and appended to the target Mailbox or Folder, the Server sends the synchronous `messageAdded` message containing the UID of the newly appended message.

Body:

> The XML presentation of the Message (the `EMail` data).
> The text parts of the message XML presentation should use the Line Feed (0x0A) symbol as the EOL (end of line) symbol.

Example:
The Client appends a simple text message to the Mailbox "Sent", with the `Seen` and `Answered` flags. If the Mailbox does not exist, it is created.

```
C:<messageAppend id="A001" targetMailbox="Sent" flags="Seen,Answered" mailboxClass="" >
    <EMail>
        <From realName="Sender Name">fromName@domain</From><Subject>I'll be there!</Subject>
        <To realName="Recipient Name">toName@domain</To><X-Mailer>SuperClient v7.77</X-Mailer>
        <Date localTime="20060621T215124" timeShift="-25200">20070830T045124Z</Date>
        <Message-ID>web-40721383@domain.dom</Message-ID>
        <MIME type="text" subtype="plain">Dear Susan,&#x0A;&#x0A;I will come to your place tomorrow, thank you for the
invitation!&#x0A;Mary.&#x0A;</MIME>
    </EMail>
  </messageAppend>
S:<response id="A001"/>
```

This operation adds the following message to the Mailbox:

```
From: "Sender Name" <fromName@domain>
Subject: I'll be there!
To: "Recipient Name" <toName@domain>
X-Mailer: SuperClient v7.77
Date: Wed, 21 Jun 2006 21:51:24 -0700
Message-ID: <web-40721383@domain.dom>
Content-Type: text/plain; charset="utf-8"

Dear Susan,

I will come to your place tomorrow, thank you for the invitation!
Mary.
```

To create messages with file attachments, put the files into the "uploaded file set" first. Then you can specify them in the MIME elements by using the `uploadID` attribute.
You can specify the `type`, `subtype`, and (for text files) `charset` attributes. If you do not explicitly specify them, they are copied from the Content-Type field of the HTTP request used to upload the file.

Example:
The client first uploads 2 files - `test.gif` (using uploadID `att01`) and `sample.pdf` (using uploadID `att02`). Then the client appends a message to the "Drafts" Folder, with the "Draft" message flag, replacing the existing message with the 578 UID.
The message contains some text in the alternative (plain and html) formats, and the uploaded files as attachments.
The client requests the Server to send the UID of the newly created message (756).
While the client was adding the new message, a different process added some other message (with UID=755) to the "Drafts" Folder Mailbox.
The client then clears the "uploaded file set", and re-syncs its state with the "Drafts" Folder.

```
C:<messageAppend id="A001" folder="Drafts" flags="Draft,Seen" replacesUID="578" report="uid" >
```

```
        <EMail>
            <From realName="Sender Name">fromName@domain</From><Subject>Text with attachments</Subject>
            <To realName="Recipient Name">toName@domain</To><X-Mailer>SuperClient v7.77</X-Mailer>
            <Date localTime="20060621T215124" timeShift="-25200">20070830T045124Z</Date>
            <Message-ID>web-40721383@domain.dom</Message-ID>
            <MIME type="multipart" subtype="mixed">
                <MIME type="multipart" subtype="alternative">
                    <MIME type="text" subtype="plain">Dear Susan,&#x0A;&#x0A;I will come to your place tomorrow, thank you for
the invitation!&#x0A;Mary.&#x0A;</MIME>
                    <MIME type="text" subtype="html">&lt;html&gt;&lt;body&gt;&lt;i&gt;Dear Susan,&lt;/i&gt;&lt;p&gt;I will come
to your place tomorrow, thank you for the invitation!&lt;p&gt;&lt;i&gt;Mary.&lt;/i&gt;</MIME>
                </MIME>
                <MIME uploadID="att01" />
                <MIME uploadID="att02" type="application" subtype="pdf" />
            </MIME>
        </EMail>
    </messageAppend>
S:<folderReport folder="Drafts" mode="notify" />
S:<messageAdded id="A001" folder="Drafts" UID="756" />
S:<response id="A001"/>
C:<clearUploaded id="A002" />
S:<response id="A002"/>
C:<folderSync id="A003" folder="Drafts"/>
S:<folderReport id="A003" folder="Drafts" index="17" UID="578" mode="removed" messages="301" unseen="0"/>
S:<folderReport id="A003" folder="Drafts" index="127" UID="755" mode="added" messages="302" unseen="1">
    <FLAGS>Recent,Drafts</FLAGS>
    <From>Other Name</From>
  </folderReport>
S:<folderReport id="A003" folder="Drafts" index="17" UID="756" mode="added" messages="303" unseen="1" >
    <FLAGS>Recent,Drafts,Seen</FLAGS>
    <From>Sender Name</From>
  </folderReport>
S:<response id="A003"/>
```

This operation adds the following message to the Mailbox:

```
From: "Sender Name" <fromName@domain>
Subject: Text with attachments
To: "Recipient Name" <toName@domain>
X-Mailer: SuperClient v7.77
Date: Wed, 21 Jun 2006 21:51:24 -0700
Message-ID: <web-40721383@domain.dom>
Content-Type: multipart/mixed; boundary="_===38330025====my.server.domain===_"

--_===38330025====my.server.domain===_
Content-Type: multipart/alternative; boundary="_===38330025-X====my.server.domain===_"


--_===38330025-X====my.server.domain===_
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: 8bit

Dear Susan,

I will come to your place tomorrow, thank you for the invitation!
Mary.

--_===38330025-X====my.server.domain===_
Content-Type: text/html; charset="utf-8"
Content-Transfer-Encoding: 8bit

<html><body><i>Dear Susan,</i><p>I will come to your place tomorrow, thank you for the invitation!<p><i>Mary.</i>
--_===38330025-X====my.server.domain===_--

--_===38330025====my.server.domain===_
Content-Type: image/gif; name="test.gif"
Content-Disposition: attachment; filename="test.gif"
Content-Transfer-Encoding: base64

R0lGODlhCgAMAPf/AP//////zP//mf//Zv//M///AP/M///MzP/Mmf/MZv/MM//MAP+Z//+Z
zP+Zmf+ZZv+ZM/+ZAP9m//9mzP9mmf9mZv9mM/9mAP8z//8zzP8zmf8zZv8zM/8zAP8A//8A
zP8Amf8AZv8AAM/8z/zMz/mcz/Zsz/M8z/AMzM/8zMzMmczMZszMM8zMAMyZ/8yZ
zMyZmcyZZsyZM8yZAMxm/8xmzMxmmcxmZsxmM8xmAMwz/8wzzMwzmcwzZswzM8wzAMwA/8wA
zMwAmcwAZswAM8wAADn//5n/zJn/mZn/Zpn/M5n/AJnM/5nMzJnMMZnMMM5nMAJmz/5mZ
zJmZmZmZ5mZAJlm/5lmzJlmmZlmZkzZpkzM5kzJkzAJkA/5kAzJkAmZkAZkA5kAJkA/5kA
zJkAmZkAZpkAM5kAAGb//2b/zGb/mWb/Zmb/M2b/AGbM/2bMzGbMmWbMZsMM2bMMGaZ/2aZ
zGaZmWaZZZm/2ZmzGmmzmmZmM2ZmAGYz/2YzzGYzmWYzZmYzM2YzAGYA/2YAzGYAmZYAZkA
zGYAM2YAAGb/AGYM/mTP/ZjP/MzP/AMzM/wDzACDMMmQDMMGzACCZ/wCZ
zACZmQCZZmcMmZmcMOzzwWBzwBmBwBmAAAZ/wAZ
zAAmQAAZgAAM+4AAN0AALsAAKoAAIgAAHcAAFUAEQAACIAABEAAADuAADdAAC7AACqAACI
AAB3AABVAABEAAAiAAARAAAA7gAA3QAAuwAAqgAAi4AAAiAAAdwAAVQAARAAIgAAEe7u7t3d3u7
u6qqoiIiHd3d1VVVURCIiIhEREQAAACH+HUdyZkJ1aWxkZXIgMC40lGJ5IF2ZXMgUGln
dWV0ACH5BAUEALkALAAAAAKAAwAAg1AHPlG0gw379cAgEoXHgw4UKFDfM9hIhO4sSIEwFg
vCiQ4L+PICl1m/Cfx34qQGkVeBMkSZEAAOw==

--_===38330025====my.server.domain===_
Content-Type: application/pdf; name="sample.pdf"
Content-Disposition: attachment; filename="sample.pdf"
Content-Transfer-Encoding: base64

JVBERi0xLjIgDSXi48/TDQogDTggMCBvYmoNPDwNL0xlbmd0aCA5IDAgUg0vRmlsdGVyIC9G
bGF0ZURlY29kZSANPj4Nc3RyZWFtDQpIic2X3VLjuBaFn6DfQXfDuaCP5X9fJiQEpkNI2eFQ
[......skipped......]
Ug0vSUQgWzxjNjIyNzFiYzY4YmFlYjJ3YzBkM2ViNTk4MjJiZTA4Nz48YzYyMjcxYmM2OGJh
ZWI2N2MwZDNlYjU5ODIyYmUwODc+XQ0+Pg1zdGFydHhyZWYNODc1NA0lJUVPRg0=

--_===38330025====my.server.domain===_--
```

To create messages with MIME parts (attachments) copied from other messages stored on the Server, use the copyMIME element instead of a MIME element:

```
copyMIME
```
Attributes:

```
folder or calendar
```
the name of an open Folder or an open Calendar.

```
UID
```
the message UID.

```
partID
```
this optional attribute specifies the message part to copy.
If it is not specified, the entire message is copied.
If it is specified, the non-MIME (other than `Content-xxxxx`) part header fields are not copied.

Example:
The Client stores a note in the "Notes" Mailbox, appending parts 3 and 4 (attachments) of the message 156 stored in the INBOX folder:

```
C:<messageAppend id="A001" targetMailbox="Notes" flags="Seen">
    <EMail>
      <From realName="Sender Name">fromName@domain</From>
      <Subject>Vacation Pictures</Subject>
      <Date localTime="20060621T215124" timeShift="-25200">20070830T045124Z</Date>
      <Message-ID>web-40721383@domain.dom</Message-ID>
      <MIME type="multipart" subtype="mixed">
        <MIME type="text" subtype="plain">The first part of the underwater shots.</MIME>
        <copyMIME folder="INBOX" UID="156" partID="3"/>
        <copyMIME folder="INBOX" UID="156" partID="4"/>
      </MIME>
    </EMail>
  </messageAppend>
S:<response id="A001"/>
```

You can attach files stored in the Personal File Storage. Specify the full file name in the MIME elements by using the `fileName` attribute.
You must explicitly specify `type`, `subtype`, and (for text files) `charset` attributes.
You should specify the `disposition-filename` attribute, as the `fileName` attribute is not used to form the attachment name.

Example:
The Client stores a note in the "Notes" Mailbox, appending the file `private/MyFile.jpg` as the `photo.jpg` attachment:

```
C:<messageAppend id="A001" targetMailbox="Notes" flags="Seen">
    <EMail>
      <From realName="Sender Name">fromName@domain</From>
      <Subject>Vacation Pictures</Subject>
      <Date localTime="20060621T215124" timeShift="-25200">20070830T045124Z</Date>
      <Message-ID>web-40721383@domain.dom</Message-ID>
      <MIME type="multipart" subtype="mixed">
        <MIME type="text" subtype="plain">Attached please find the images I have stored as files.</MIME>
        <MIME type="image" subtype="jpeg" fileName="private/MyFile.jpg" disposition-filename="photo.jpg" />
      </MIME>
    </EMail>
  </messageAppend>
S:<response id="A001"/>
```

Note: if a `<MIME />` element has the `disposition` attribute value `none`, the `Content-Disposition:` header field is not created, and the `Content-Type` field is stored without the `name` parameter.

Note: the text MIME parts should use a single LineFeed (0x0A, decimal 10) symbol as the line separator.

You can ask the Server to use the "flowed" format for text MIME parts.

Example:

```
C:<messageAppend id="A001" targetMailbox="Notes" flags="Seen">
    <EMail>
      <Subject>Test text</Subject>
      <MIME type="text" subtype="plain" format="flowed">This is a very long long long long long long long long long
long superlong long long long long long long line, followed with a shorter line:
This is a shorter line.</MIME>
    </EMail>
  </messageAppend>
S:<response id="A001"/>
```

This operation adds the following message to the Mailbox:

```
Subject: Test text
Content-Transfer-Encoding: 8bit
Content-Type: text/plain; format="flowed"; charset="utf-8"

This is a very long long long long long long long long long superlong
long long long long long long line, followed with a shorter line:
This is a shorter line.
```

(there is a trailing space after the "superlong" word)

To create a signed message, make sure that S/MIME Private key is unlocked, and specify the `sign` attribute in the topmost `EMail` element.

```
C:<messageAppend id="A001" targetMailbox="Sent" flags="Seen,Answered" >
    <EMail sign="yes">
        <From realName="Sender Name">fromName@domain</From><Subject>I'll be there!</Subject>
        <To realName="Recipient Name">toName@domain</To><X-Mailer>SuperClient v7.77</X-Mailer>
        <MIME type="text" subtype="plain">Dear Susan,&#x0A;&#x0A;I will come to your place tomorrow, thank you for the
invitation!&#x0A;Mary.&#x0A;</MIME>
    </EMail>
  </messageAppend>
S:<response id="A001"/>
```

This operation adds the following message to the Mailbox:

```
From: "Sender Name" <fromName@domain>
Subject: I'll be there!
To: "Recipient Name" <toName@domain>
X-Mailer: SuperClient v7.77
Content-Type: multipart/signed; protocol="application/x-pkcs7-signature"; micalg="SHA1";
boundary="_===signed==2610002====my.server.domain===_"

This is a signed S/MIME message

--_===signed==2610002====my.server.domain===_
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: 8bit

Dear Susan,

I will come to your place tomorrow, thank you for the invitation!
Mary.

--_===signed==2610002====my.server.domain===_
Content-Type: application/x-pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"

MIIE6wYJKoZIhvcNAQcCoIIE3DCCBNgCAQExCzAJBgUrDgMCGgUAMAsGCSqGSIb3DQEHAaCC
ArkwggK1MIICX6ADAgECAgcAk9RF+n/7MA0GCSqGSIb3DQEBBAUAMIG6MQswCQYDVQQGEwJV
UzELMAkGA1UECBMCQ0ExFDASBgNVBAcTC01pbGwgVmFsbGV5MSIwIAYDVQQKExlDb21tdW5p
.............
gIbXNT64QJ+gEYkI9mnePiS1TUOOzGYfXaLy1pqm6jmzBUt7/3UY8ZNHVIwM0Fzj7NwzqM1U
Esbkyi3WHNxTZ4HSCs8J2enGQEZjNWHOuX96xQojYGLV0m5Z/FatV9GQ8jNVBmQ9xYGKxmlY
jT9ze/oHyKuj7KR8QrgQSYiJVnn7

--_===signed==2610002====my.server.domain===_--
```

`messageSubmit`

This operation tells the Server to compose a Message and submit it to the Queue for delivery. A message copy can be stored in a Mailbox.

Attributes:
`useDSN`

> if this optional attribute exists, and its value is `yes`, delivery reports are generated when a message is delivered, or if delivery fails.
>
> If this attribute is absent, delivery reports are generated only when message delivery fails.

`targetMailbox`, `mailboxClass`, `flags`, `internalDate`

> if the optional `targetMailbox` attribute is specified, then the composed message is appended to the specified Mailbox. These attributes have the same meaning as for the `messageAppend` operation.

`dataset`

> if the optional attribute is specified, it specifies the name of an Account dataset. All message target addresses (see below) are stored in the specified dataset, removing existing duplicates.

Body:

> The [XML presentation](#) of the Message (the `EMail` element), and zero, one, or several `Envelope-To` elements. Each `Envelope-To` element should have a text body - the E-mail address to send the message to.
> If no `Envelope-To` element is specified, the message is sent to the addresses specified in all addresses specified in the `To`, `Cc`, and `Bcc` elements inside the `EMail` element.
> If the `EMail` element contains the `uploadID` attribute, the RFC message body is built from the contents of the

respective file in the session "uploaded file set".

When the message MIME text is generated, all addresses from the `Bcc` elements are skipped.

If the XML presentation does not include the `Message-ID`, `Date`, `MIME-Version` fields, the server creates these message fields itself.

Example:

The Client sends a simple text message to the `toName@domain address` and the Bcc copy is sent to `bccName@domain` address. The Client requests delivery notification.

```
C:<messageSubmit id="A001" useDSN="yes" >
    <EMail>
       <From realName="Sender Name">fromName@domain</From>
       <Subject>I'll be there!</Subject>
       <To realName="Recipient Name">toName@domain</To>
       <X-Mailer>SuperClient v7.77</X-Mailer>
       <Bcc>bccName@domain</Bcc>
       <MIME type="text" subtype="plain">Dear Susan,&#x0A;&#x0A;I will come to your place tomorrow, thank you for the
invitation!&#x0A;Mary.&#x0A;</MIME>
    </EMail>
  </messageSubmit>
S:<response id="A001"/>
```

Example:

The Client sends a simple text message to the `a1@domain` and `a2@domain` addresses (different from the addresses specified in the `To` and `Cc` elements).

The message copy should be stored in the "Sent Items" Mailbox with the "Seen" flag.

The Client requests delivery notification.

```
C:<messageSubmit id="A001" targetMailbox="Sent Items" flags="Seen" >
    <Envelope-To>a1@domain address</Envelope-To>
    <Envelope-To>a2@domain address</Envelope-To>
    <EMail>
       <From realName="Sender Name">fromName@domain</From>
       <Subject>I'll be there!</Subject>
       <To realName="Recipient Name">toName@domain</To>
       <X-Mailer>SuperClient v7.77</X-Mailer>
       <Cc>ccName@domain</Cc>
       <MIME type="text" subtype="plain">Dear Susan,&#x0A;&#x0A;I will come to your place tomorrow, thank you for the
invitation!&#x0A;Mary.&#x0A;</MIME>
    </EMail>
  </messageSubmit>
S:<response id="A001"/>
```

Example:

The Client forwards the message 156 stored in the INBOX folder to the `a1@domain` address:

```
C:<messageSubmit id="A001">
    <EMail>
       <From realName="Sender Name">fromName@domain</From>
       <Subject>Fwd: Sorry, I cannot make it :-(</Subject>
       <To realName="Recipient Name">a1@domain</To>
       <X-Mailer>SuperClient v7.77</X-Mailer>
       <MIME type="multipart" subtype="mixed">
         <MIME type="text" subtype="plain">Dear Susan,&#x0A;&#x0A;Attached please find the message I received this
morning...&#x0A;Mary.&#x0A;</MIME>
         <copyMIME folder="INBOX" UID="156" />
    </EMail>
  </messageSubmit>
S:<response id="A001"/>
```

This operation adds the following message to the Mailbox:

```
From: "Sender Name" <fromName@domain>
Subject: Fwd: Sorry, I cannot make it :-(
To: "Recipient Name" <toName@domain>
X-Mailer: SuperClient v7.77
Date: Wed, 21 Jun 2006 22:55:24 -0800
Message-ID: <ximss-38150012@this.server.dom>
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="_===38330025====my.server.domain===_"

--_===38330025====my.server.domain===_
Content-Type: text/plain; charset="utf-8"

Dear Susan,

Attached please find the message I received this morning...
Mary.

--_===38330025====my.server.domain===_
Content-Type: message/rfc822

From: "Barbara Smith" <barbara@domain>
Subject: Sorry, I cannot make it :-(
To: "Sender Name" <fromName@domain>
X-Mailer: OtherClient v8.88
```

```
Date: Wed, 21 Jun 2006 21:51:24 -0800
Message-ID: <zzzzzzzzz@other.server.dom>
Content-Type: text/plain; charset="utf-8"

Mary,

Sorry, but I will be out of town tomorrow...

Barbara.

--_===38330025====my.server.domain===_--
```

**messageRedirect**

This operation tells the Server to compose a copy of a message stored in a Mailbox, and to submit it to the Queue for delivery.

Attributes:

`folder`
> the Folder name.

`UID`
> the message UID.

`partID`
> this optional attribute specifies the message part to read. If it is not specified, the entire message is redirected. If specified, it should specify a message part that is an E-mail message: a sub-part of the `message/rfc822` part, a digest part, etc.

`mirror`
> if this optional attribute is `yes`, then the message is "mirrored": it is sent to the specified addresses, but the message header `To` and `Cc` fields are left unchanged.

`markRedirected`
> if this attribute is `yes`, then the "Redirected" flag is added to the message.
> If this attribute is not specified, the `yes` value is assumed if the `partID` is not specified, otherwise the `no` value is assumed.

Body:
> A set of one or more `To` elements specifying the recipient address(es).

Example:
The Client redirects the message 156 stored in the INBOX folder to the `a1@domain` and `a2@domain` addresses:

```
C:<messageRedirect id="A001" folder="INBOX" UID="156">
    <To realName="Recipient Name">a1@domain</To>
    <To realName="Recipient-2">a2@domain</To>
  </messageRedirect>
S:<response id="A001"/>
```

**messageForward**

This operation tells the Server to compose a copy of a message stored in a Mailbox, and to submit it to the Queue for delivery. The operation uses the same attributes as the `messageRedirect` operation, but the `mirror` attribute is not supported.

**messageConfirm**

This operation tells the Server to compose an MDN (Message Disposition Notification) report for a message stored in a Mailbox, and to submit it to the Queue for delivery.
Client applications should use this operation when:

- the message has been displayed to the user, and
- the message has the Disposition-Notification header field, and
- the message does not have the $MDNSent [Message Flag](#), and
- the user Preference `SendMDNMode` is set to `Automatically` or it is set to `Manually` and the user has explicitly requested to send a confirmation.

Attributes:
`folder`

the Folder name.

UID
> the message UID.

type
> this attribute should be set to `auto` (if the confirmation is generated automatically) or to `manual` (if the user has explicitly requested to send a confirmation).

`messageAttrRead`
> This operation reads the Mailbox message attributes.
>
> Attributes:
> folder
>> the Folder name.
>
> UID
>> the message UID.
>
> Body:
>> a set of `field` XML elements. Each element should have a text body containing the name of an attribute to retrieve.
>
> If no `field` element is specified, all attributes are retrieved.
> The Server returns a `messageAttrData` message.

`messageAttrWrite`
> This operation modifies Mailbox message attributes.
>
> Attributes:
> folder
>> the Folder name.
>
> UID
>> the message UID.
>
> Body:
>> XML presentation of a dictionary with new attribute values. To remove an attribute specify the null-object (`<null/>`) value.

The Server sends the following data messages:

`folderMessage`
> This synchronous data message is sent when the Server processes the <u>folderRead</u> request.
>
> Attributes:
> folder, UID, partID, mode
>> copies of the <u>folderRead</u> request attributes.
>
> Body:
>> The <u>XML presentation</u> of the Message or its part.

`messageAdded`
> These synchronous data messages are sent when the Server processes the <u>messageAppend</u>, <u>contactAppend</u>, and <u>messageCopy</u> requests with the `report` attributes set.

Attributes:

`folder` or `targetMailbox`
>    the target Folder or Mailbox name, copied from the request attribute.

`UID`
>    the newly added message UID.

**Note:** even if this data message is sent for a Folder, the Server will still send asynchronous `folderReport` data message notifying the client about a new message added to the Folder. The client SHOULD use the `folderSync` operation to synchronize its internal view with the Server view, otherwise the newly added message will not be "seen" in the Folder.

`messageAttrData`
>    These synchronous data messages are sent when the Server processes the `messageAttrRead` requests.

Attributes:

`folder`, `UID`
>    copied from the request.

Body:
>    The XML presentation of the attribute set dictionary.

---

# Account Management

A client can use the following set of operations to manage Account ACLs.

`accountRightsGet`
>    This operation makes the Server send an `accountRights` data message (see below) with the user access rights for the current or specified Account.

Attributes:

`userName`
>    an optional account specifying the target Account name. If this attribute is absent, the current Account is used.

`accountACLList`
>    This operation makes the Server send an `accountACL` data message (see below) with the Mailbox Access Control List data.

Attributes:

`userName`
>    an optional account specifying the target Account name. If this attribute is absent, the current Account is used.

`accountACLUpdate`
>    This operation modifies the Account Access Control List.

Attributes:

`userName`
>    an optional account specifying the target Account name. If this attribute is absent, the current Account is used.

Body:
>    A set of `aclElem` XML elements. These elements are the same as those used with the `mailboxACLUpdate` operation, but their body strings use symbols that specify Account Access Rights.

The Server sends the following data messages:

`accountRights`
> This message is sent when the Server processes the <u>`accountRightsGet`</u> request.
>
> Attributes:
> `userName`
>> the Account name (if specified in the request).
>
> Body:
>> a string; each symbol specifies an effective <u>Account Access Right</u> granted to the current user.

`accountACL`
> This message is sent when the Server processes the <u>`accountACLList`</u> request.
>
> Attributes:
> `userName`
>> the Account name (if specified in the request).
>
> Body:
>> A set of `aclElem` XML elements, one element for each Account Access Control List element, same as used in the <u>`mailboxACL`</u> data messages, but the element body strings contain symbols representing <u>Account Access Rights</u>.

---

# Secure Messaging (S/MIME)

A client can use the following set of operations to use the Server S/MIME features.

`SMIMEUnlock`
> This operation unlocks the Account Private Key stored on the Server.
> When the Private Key is unlocked, the Server decrypts encrypted parts of messages retrieved using the `folderRead` operation.
> The Server can also encrypt and/or digitally sign submitted messages.
> **Note:** this operation transfers the unlocking string ("storage password") in clear text. The client application should use this command only over a secure (TLS) connection.
> **Note:** the unlocked Private Key is added to the current session data only, and it is not stored anywhere. If a different session is opened for the same Account, it should unlock the Private Key by itself.
>
> Attributes:
> `password`
>> the unlocking string (password).
>
> `duration`
>> optional attribute - the time (in seconds) to keep the Private Key unlocked; the Private Key is automatically locked after this period of time.

`SMIMELock`
> This operation locks the Account Private Key stored on the Server (it removes the unlocked Private Key from the current session data).
> It is recommended to use this operation after a certain period of user inactivity.

`SMIMEModifyPassword`

This operation changes the password string protecting the Account Private Key in the Server storage. The Account Private Key must be unlocked.

**Note:** this operation transfers the unlocking string ("storage password") in clear text. The client application should use this command only over a secure (TLS) connection.

Attributes:

`password`
  the new unlocking string (password).

`SMIMEGet`
  This operation retrieves S/MIME-related information stored on the Server.

Attributes:

`type`
  type of information retrieved.

If the value of the `type` attribute is `certificate`, then the Server sends a `certificate` data message with the Account Public Certificate.

`SMIMESet`
  This operation modifies S/MIME-related information stored on the Server.

Attributes:

`mode`
  operation: `add`, `new`, `update`, `delete`.

`password`
  the S/MIME unlocking password (required for the `add` and `new` operations).

`filePassword`
  the password used to decrypt the imported data (required for the `add` and `delete` operations).

`duration`
  when a new S/MIME key and certificate are set, they remain unlock. If this attribute is set, it specifies the number of seconds the S/MIME data remains unlock.

`PFX`
  this attribute is used only if the body does not contain a `PFX` XML element. This attribute identifies a file in the "uploaded file set". The file should contain PKCS12 data containing a Private Key and a Certificate.

Body:
  an optional `PFX` element; it should contain a `base64` XML element with a textual body that is a Base64-encoded PKCS12 data containing a Private Key and a Certificate.

If the `mode` value is `add`, the Server gets the PKCS12 data specified with the `PFX` body XML element or with the `PFX` attribute, decrypts it using the specified `filePassword` value, and assigns the decrypted Private Key and Certificate to the current Account.
The server encrypts the stored Private Key data using the the specified `password` value, which becomes the "S/MIME unlocking string".

If the `mode` value is `new`, the Server generates a random Private Key and issues an S/MIME Certificate for the current Account, using the Account Domain "S/MIME Issuer" Certificate.
The server encrypts the stored Private Key data using the the specified `password` value, which becomes the "S/MIME unlocking string".

If the `mode` value is `update`, the Server updates the Account Certificate, if this Certificate was issued using the Account Domain "S/MIME Issuer" Certificate.

If the `mode` value is `delete`, the Server gets the PKCS12 data specified with the `PFX` body XML element or with the `PFX`

attribute, decrypts it using the specified `filePassword` value, and compares the retrieved Private Key with the Private Key of the current Account.
If the Private Keys are the same, the Account Private Key and Certificate are removed.

**Note:** this operation transfers the storage passwords in clear text. The client application should use this command only over a secure (TLS) connection.

The Server sends the following data messages:

`certificate`
This message is sent when the Server processes the <u>SMIMEGet</u> request.

Body:
an XML presentation of the Account S/MIME Public <u>Certificate</u>.

The Server detects all signed message parts and tries to verify that the part body has not been altered since the time it was signed, and that the signature certificates are valid (i.e. issued by known authorities).

The `MIME` element body for a signed message part contain 2 XML sub-elements: the first element is the signed data (an `EMail` or `MIME` element), the second part is the `SMIMESignature` XML element.

If the Server failed to decode or decrypt the signature data, the `MIME` element body for a signed message part contains the `decryptionError` attribute with the error code text.

The `SMIMESignature` element body contains Certificate sub-elements for all signatures that match the part data. If the Server failed to verify the Certificate, its element contains the validationError attribute with the error code text. If no signature matches the the part data, the `SMIMESignature` element is empty.

Example:
The Client retrieves a signed message:

```
C:<folderRead id="A001" folder="INBOX" UID="55" totalSizeLimit="100000" />
S:<folderMessage id="A001" folder="INBOX" UID="55">
  <EMail>
    <Return-Path>fromName@domain</Return-Path>
    <From realName="Sender Name">fromName@domain</From>
    <Subject>Re: I'll be there!</Subject>
    <To realName="Recipient Name">a1@domain</To>
    <X-Mailer>SuperClient v7.77</X-Mailer>
    <Date localTime="20070830T204318" timeShift="-25200">20070830T184318Z</Date>
    <Message-ID>web-40721383@domain.dom</Message-ID>
    <MIME digesterName="SHA1" estimatedSize="5171" subtype="signed" type="multipart"
        Type-micalg="SHA1" Type-protocol="application/x-pkcs7-signature" />
      <MIME estimatedSize="3032" partID="01" subtype="mixed" type="multipart">
        <MIME charset="utf-8" estimatedSize="86" partID="01-01" subtype="plain" type="text">Dear Susan,&#x0A;&#x0A;Attached
please find a file I received this morning...&#x0A;Mary.&#x0A;
        </MIME>
        <MIME disposition="attachment" Disposition-filename="logo.gif" estimatedSize="1929" partID="01-02" subtype="gif"
type="image" />
      </MIME>
      <SMIMESignature>
        <x509 subject="fromName@domain" validationError="presented certificate is issued by an unknown authority"
version="2">
          <subject><cn>Sender Name</cn><contact>fromName@domain</contact></subject>
          <issuer><c>US</c><cn>issuer.dom</cn><contact>postmaster@issuer.dom</contact><l>Mill Valley</l>
              <o>Issuer Company, Inc.</o><ou>Issuer Department</ou><st>CA</st></issuer>
          <validFrom>20040924T231857Z</validFrom>
          <validTill>20170923T231857Z</validTill>
        </x509>
      </SMIMESignature>
    </MIME>
  </EMail>
  </folderMessage>
S:<response id="A001"/>
```

When the Account Private Key is unlocked, the Server tries to decrypt all encrypted message parts.

If decrypting fails, the encrypted part element contains an additional attribute:
`decryptionError`
the decrypting error message text.

If decrypting fails, the encrypted part is shown as the MIME or EMail XML sub-element of the encrypted part element.

If decrypting succeeds, the encrypted part element body contains a MIME or EMail XML sub-element with the decrypted data.

The encrypted part element contains additional attributes:
cipherName
      the name of cryptographic cipher used to encrypt the decrypted part.
keyLength
      the length of cryptographic cipher key (in bits) used to encrypt the decrypted part.

Example:
The Client retrieves an encrypted message, then unlocks the Account Private Key and then retrieves the same message again:

```
C:<folderRead id="A001" folder="INBOX" UID="55" totalSizeLimit="100000" />
S:<folderMessage id="A001" folder="INBOX" UID="55">
   <EMail>
     <Return-Path>fromName@domain</Return-Path>
     <From realName="Sender Name">fromName@domain</From>
     <Subject>Re: I'll be there!</Subject>
     <To realName="Recipient Name">a1@domain</To>
     <X-Mailer>SuperClient v7.77</X-Mailer>
     <Date localTime="20070830T204318" timeShift="-25200">20070830T184318Z</Date>
     <Message-ID>web-40721383@domain.dom</Message-ID>
     <MIME disposition="attachment" Disposition-filename="smime.p7m" estimatedSize="4536"
           subtype="x-pkcs7-mime" type="application" Type-name="smime.p7m" Type-smime-type="enveloped-data" />
   </EMail>
   </folderMessage>
S:<response id="A001"/>
C:<SMIMEUnlock id="A002" password="smime-password" />
S:<response id="A002"/>
C:<folderRead id="A003" folder="INBOX" UID="55" totalSizeLimit="100000" />
S:<folderMessage id="A003" folder="INBOX" UID="55">
   <EMail>
     <Return-Path>fromName@domain</Return-Path>
     <From realName="Sender Name">fromName@domain</From>
     <Subject>Re: I'll be there!</Subject>
     <To realName="Recipient Name">a1@domain</To>
     <X-Mailer>SuperClient v7.77</X-Mailer>
     <Date localTime="20070830T204318" timeShift="-25200">20070830T184318Z</Date>
     <Message-ID>web-40721383@domain.dom</Message-ID>
     <MIME cipherName="RC2" disposition="attachment" Disposition-filename="smime.p7m" estimatedSize="4536" keyLength="40"
           subtype="x-pkcs7-mime" type="application" Type-name="smime.p7m" Type-smime-type="enveloped-data" >
       <EMail partID="E">
         <From realName="Sender Name">fromName@domain</From>
         <Subject>Re: I'll be there!</Subject>
         <To realName="Recipient Name">a1@domain</To>
         <X-Mailer>SuperClient v7.77</X-Mailer>
         <Date localTime="20070830T204318" timeShift="-25200">20070830T184318Z</Date>
         <Message-ID>web-40721383@domain.dom</Message-ID>
         <MIME estimatedSize="3275" partID="E" subtype="mixed" type="multipart">
           <MIME charset="utf-8" estimatedSize="329" partID="E-01" subtype="plain" type="text" Type-format="flowed">
             This is an encrypted E-mail with an attachment.

             On Thu, 30 Aug 2007 20:40:49 -0800
             &quot;Sender Name&quot; &lt;fromName@domain&gt; wrote:
             &gt; Dear Susan,
             &gt;
             &gt; I will come to your place tomorrow, thank you for the invitation!
             &gt; Mary.
           </MIME>
           <MIME disposition="attachment" Disposition-filename="logo.gif" estimatedSize="1929" partID="E-02"
                 subtype="gif" type="image" />
         </MIME>
       </EMail>
     </MIME>
   </EMail>
</folderMessage>
S:<response id="A003"/>
```

When the Account Private Key is unlocked, a client can submit a signed E-mail.

The EMail element should contain an additional attribute:
sign
      if this optional attribute exists, and its value is yes, the message body is signed.

Example:
The Client submits a signed text message to the toName@domain address.

```
C:<messageSubmit id="A001">
   <EMail sign="yes">
     <From realName="Sender Name">fromName@domain</From>
     <Subject>I'll be there!</Subject>
     <To realName="Recipient Name">toName@domain</To>
     <X-Mailer>SuperClient v7.77</X-Mailer>
     <MIME type="text" subtype="plain">Dear Susan,&#x0A;&#x0A;I will come to your place tomorrow, thank you for the
invitation!&#x0A;Mary.&#x0A;</MIME>
   </EMail>
   </messageSubmit>
S:<response id="A001"/>
```

When the Account Private Key is unlocked, a client can submit an encrypted E-mail if there is an Address Book containing records for all E-mail recipients, and each of those records contains a recipient PKI Certificate.

The topmost `EMail` element should contain additional attributes:

`encrypt`
    if this optional attribute exists, and its value is `yes`, the message body is encrypted.

`addressBook`
    the name of the Address Book to look the recipient certificates in.

`addressBook1`, `addressBook2`
    optional: names of the additional Address Books. If no recipient certificate is found in the Address Book with the name stated in the `addressBook` attribute, these Address Books are checked.

The topmost `EMail` element should contain the single "inner" `EMail` element - the content of that element will be encrypted. The inner `EMail` element header fields can be the same as in the topmost `EMail` element, or they can be different. For example, the "inner" `EMail` element `Subject` element can be different.

**Note:** you cannot specify both the `sign` and `encrypt` attributes for the topmost `EMail` element. To send an encrypted and signed message, add the `sign` attribute to the inner `EMail` element.

Example:
The Client submits an encrypted and signed text message to the `toName@domain` address. Note unmatching subjects. The Client requests delivery notification.

```
C:<messageSubmit id="A001" useDSN="yes" >
    <EMail encrypt="yes" addressBook="Contacts" >
       <From realName="Sender Name">fromName@domain</From>
       <Subject>A message regarding your request</Subject>
       <To realName="Recipient Name">toName@domain</To>
       <EMail sign="yes">
         <From realName="Sender Name">fromName@domain</From>
         <Subject>I'll be there!</Subject>
         <To realName="Recipient Name">toName@domain</To>
         <X-Mailer>SuperClient v7.77</X-Mailer>
         <MIME type="text" subtype="plain">Dear Susan,&#x0A;&#x0A;I will come to your place tomorrow, thank you for the
invitation!&#x0A;Mary.&#x0A;</MIME>
       </EMail>
    </EMail>
  </messageSubmit>
S:<response id="A001"/>
```

# Contacts Operations

Contacts (vCard and vCardGroup) information can be stored as an E-mail item in any Mailbox. A vCard data element can be attached (as a MIME part) to any E-mail message. Each message can contain several MIME parts each containing several vCard objects.

MIME parts with the `text` type and `x-vcard` or `dictionary` subtype contain vCard elements.

MIME parts with the `text` type and `x-vgroup` subtype contain one vCardGroup element.

To include vCard data into a message (using the `messageAppend`, `messageSubmit` operations) include a MIME element with `type="text"` and `subtype="directory"` attributes. The element body should contain one or more vCard elements.

To include vCardGroup data into a message (using the `messageAppend`, `messageSubmit` operations) include a MIME element with `type="text"` and `subtype="x-vgroup"` attributes. The element body should contain one vCardGroup element.

The Contact-class Mailboxes are used to store special messages ("items") containing only vCard or vCardGroup data. In a properly composed vCard item:

- the `Subject` field contains the vCard `FN` property.
- the `To` field contains the vCard `EMAIL` properties.
- the `X-Telnum` field contains the vCard `TEL` properties.
- the `X-Has-Certificate` field exists and contains the `true` value if the vCard contains a `KEY` property.

In a properly composed vCardGroup item:

- the `Subject` field contains the vCardGroup `NAME` property.

Use the following operations to compose, store, and retrieve these special messages:

**contactAppend**

> This operation composes a special E-mail message containing the vCard or vCardGroup data as its body, forms all necessary E-mail message header fields and stores the resulting message in the specified Folder or Mailbox.
>
> Attributes:
> **folder**
>> the target Folder name. If specified, the `targetMailbox` attribute is ignored.
>
> **targetMailbox**
>> the target Mailbox name. It must be specified if the `folder` attribute is absent.
>> if the target Mailbox does not exist, it is created.
>
> **flags, replacesUID, replaceMode, report**
>> these optional attributes have the same meaning as the same `messageAppend` attributes. If the `flags` attribute is not specified, its value assumed to be `Seen`.
>
> Body:
>> exactly one [vCard](#) or [vCardGroup](#) element.
>
> The operation adds the vCard `UID` and `FN` attributes if they are missing.
>
> Example 1:
> The Client appends a vCard object to the `Contacts` Mailbox.

```
C:<contactAppend id="A011" targetMailbox="Contacts" >
    <vCard>
      <NAME>Bjorn Jensen</NAME>
      <N><FAMILY>Jensen</FAMILY><GIVEN>bjorn</GIVEN>
        <MIDDLE>A</MIDDLE><PREFIX>Mr.</PREFIX><SUFFIX>II</SUFFIX></N>
      <EMAIL><INTERNET /><USERID>bjorn@domain.dom</USERID></EMAIL>
      <TEL><WORK /><VOICE /><MSG /><NUMBER>+1 313 747-4454</NUMBER></TEL>
      <KEY><x509 /><BINVAL>dGhpcyBjb3VsZCBiZSAKbXkgY2VydGlmaWNhdGUK</BINVAL></KEY>
    </vCard>
  </contactAppend>
S:<response id="A011"/>
```

**contactsImport**

> This operation parses an uploaded file, which should contain vCard items. The resulting items are copied into the specified Folder.
>
> Attributes:
> **folder**
>> the Folder name.
>
> **uploadID**
>> a string identifying a file in the "uploaded file set".

**contactFind**

> This operation searches the open Folder for vCard items (messages) with an E-mail or Telephone number matching the specified address.
> When a "fuzzy" search is requested, the operation searches for vCard items containing E-mail addresses or "File as" names that include the specified address as a substring.
> Only the E-mail message header fields (`To` and `X-Telnum`) are checked, not the actual vCard data.
>
> Attributes:
> **folder**
>> the Folder name.
>
> **totalSizeLimit**
>> same as for the `folderRead` operation. If set to zero, no vCard message body is returned.

peer
> the address to search for.

limit
> the optional parameter limiting the number of items to find. The operation finishes when it finds the specified number of matching items or when all folder items are checked.

mode
> if this optional attribute is specified and it has the `sub` value, "fuzzy search" is requested.

The found vCard message content is sent to the client as the `folderMessage` data message, with the additional attributes:

Attributes:

folder, peer
> the same values as in the request.

UID
> the found vCard message UID (numeric).

foundAddress
> optional; the `To` or `X-Telnum` address element that matches the `peer` value.

peerName
> optional; the "File as" name of found vCard.

---

# Calendar Operations

A client can use the following operations to process a Calendar in the authenticated Account, as well as in other Accounts (by specifying the full Mailbox name as *~accountName@domainName/mailboxName*).

calendarOpen
> This operation opens the specified Mailbox as a "Calendar".
> A Calendar represents a Server Mailbox, with all messages being parsed and all calendaring information retrieved.
> Alternatively, a Calendar can represent an iCalendar document (a set of iCalendar events) retrieved via the HTTP/HTTPS protocol using the specified URL.
> Each calendar has a name, and one session cannot have two calendars with the same name. On the other hand, the same session can open the same Mailbox as two different calendars (with different names).
>
> Attributes:
>
> calendar
> > the name for the new Calendar to be opened. A client can use an arbitrary string as a Calendar name.
>
> url
> > if specified, the URL of a remote iCalendar object. The object is retrieved, parsed, and all its VEVENT elements are used as the Calendar object data. No Server Mailbox is read in this case.
>
> mailbox
> > the Mailbox name. This attribute is used only if the `url` attribute is not specified. If this attribute is not specified, the calendar name is used.
>
> A session can use several open Calendars at the same time.

calendarClose
> This operation closes an open Calendar.

Attributes:

calendar

    the Calendar name.


**findEvents**

    This operation retrieves the VEVENT objects from the specified Calendar. Only the Events that fall into the specified time interval are retrieved.

    Attributes:

calendar

    the Calendar name.

timeFrom, timeTill

    the beginning and the end of the time interval (time values should be specified for the selected time zone).

byAlarm

    if this optional attribute exists, and its value is `yes`, the operation looks not for the Events in the specified time interval, but for the Events that have an Alert element set within the specified time interval.

limit

    this optional numeric attribute limits the number of the Events returned.

skip

    this optional numeric attribute specifies how many "to be returned" Events should be skipped. Using this attribute, a client can retrieve a large Event set in smaller "chunks".


    The Server sends one <u>events</u> data message for each 24-hour day (in the selected time zone) included into the specified time interval.

**findTasks**

    This operation retrieves the VTODO objects from the specified Calendar. Only the Tasks that fall into the specified time interval are retrieved.

    Attributes:

calendar, timeFrom, timeTill, byAlarm, limit, skip

    these attributes have the same meanings as the `findEvents` operation attributes.


    The Server sends one <u>tasks</u> data message for each 24-hour day (in the selected time zone) included into the specified time interval.

**calendarRead**

    This operation tells the Server to retrieve a Message or its part. It is sent to the client as a <u>calendarMessage</u> data message. This operation is the same as the <u>folderRead</u> operation, but instead of the `folder` attribute it uses the `calendar` to specify the open Calendar name.
    **Note:** this operation cannot be applied to a Calendar object built using a remote (URL-specified) iCalendar object. Use the `calendarReadItem` operation instead.

**calendarReadItem**

    This operation tells the Server to retrieve a calendar item. It is sent to the client as a <u>calendarItem</u> data message.

    Attributes:

calendar

    the Calendar name.

UID

    the item message UID (a number).


**calendarPublish**

    This operation places a calendaring item into a Calendar. The existing items(s) with the same UID are removed.

Attributes:

`calendar`
> the Calendar name.

`sendRequests`
> if this optional attribute exists and its value is `no`, the item is stored without notifying participants.
> Otherwise, if there was an existing item with the same UID, Cancel requests are sent to all participants existing in the old item, but excluded from the new item.
> A meeting or task request is sent to all participants, or, if this attribute value is `new`, to all newly added participants.

`copyExceptions`
> if this optional attribute value is not `no` and the item does not contain the `recurrenceID` element, and the Calendar already contains an item with the same UID, all recurrence exceptions from the existing item are copied into the newly published item.

Body:
> An [iCalendar](#) element to place into the selected Calendar, and optional [MIME](#) and/or [copyMIME](#) elements to add (the same as elements used with the `messageAppend` operation).
> If no `MIME` or `copyMIME` element is present, and there is an existing Calendar item with the same UID, the attachments are copied from that item. To prevent this, include an empty `noMIME` element.

The `iCalendar` element may contain optional [vtimezone](#) elements, and it should contain exactly one calendaring item element.
It is not required to include a `vtimezone` element for a time zone used in a calendaring item element if this time zone is one of the standard time zones known to the Server.
It is allowed to omit a time zone name in the properties containing date values without the `Z` suffix. In this case the time zone selected as the current user Preference TimeZone value is used.
If the item does not contain an UID element, the Server generates a unique UID and adds it to the item.
To create or update a recurrence exception item, include the `recurrenceID` element into the `vevent` element.

Example 1:
The Client published an iCalendar object to the `Calendar` calendar.

```
C:<calendarPublish id="A021" calendar="Calendar">
    <iCalendar xmlns="urn:ietf:params:xml:ns:xcal">
     <vCalendar method="PUBLISH" prodid="CommuniGate Pro 5.1.7" version="2.0">
      <vevent>
       <organizer CN="Big Boss">MAILTO:boss@company.dom</organizer>
       <attendee CN="Small Boy">MAILTO:boy@company.dom</attendee>
       <rrule>FREQ=WEEKLY;BYDAY=MO,TH</rrule>
       <dtstamp>20061022T091143Z</dtstamp>
       <uid>18927897984@kjjkjkjk-123444</uid>
       <sequence>0</sequence>
       <summary>Report Meeting</summary>
       <dtstart tzid="NorthAmerica/Pacific">20060515T100000</dtstart>
       <dtend tzid="NorthAmerica/Pacific">20060515T110000</dtend>
       <busystatus>BUSY</busystatus>
       <last-modified>20060516T034850Z</last-modified>
       <created>20060516T034850Z</created>
       <priority>5</priority>
       <description>A twice-a-week meeting to discuss the progress of the assigned projects.</description>
      </vevent>
     </vCalendar>
    </iCalendar>
    <MIME uploadID="att01" />
    <copyMIME folder="INBOX" UID="156" partID="3"/>
  </calendarPublish>
S:<response id="A021"/>
```

**Note:** If a time value is specified without the "Z" suffix, it is assumed to be a local time in the Time Zone selected for the current user.

`calendarCancel`
> This operation removes a calendaring item from a Calendar. Cancel requests are sent to all participants.

Attributes:

`calendar`
> the Calendar name.

`UID`
> optional; the message UID (numeric).

itemUID
>    optional; the Event item (iCalendar) UID string.

recurrenceId
>    optional; the recurrence ID timestamp.

sendRequests
>    if this optional attribute is `no`, cancel request messages are not sent to the item attendees.

>    Body:

>    - an optional [iCalendar] element to cancel.
>      If this element is not specified, the item with the specified iCalendar Event UID or the specified message UID is taken from the Calendar. Then all items with the same Event item (iCalendar) UID are removed.
>    - optional `requestComment` element containing a text body - a comment to include into the cancel request message.

To cancel only one recurrence of a recurrent event either specify an `iCalendar` element with the `recurrenceId` sub-element, or, if the `iCalendar` element is not specified, use the `recurrenceId` attribute.

calendarUpdate
>    This operation updates a calendaring item in a Calendar using a reply-type [iCalendar] object. This item specifies if a particular attendee has accepted or rejected the invitation.

>    Attributes:
>    calendar
>    >    the Calendar name.

>    Body:
>    >    The [iCalendar] reply-type item.

calendarForward
>    This operation tells the Server to compose a copy of an Event stored in a Calendar, and to submit it to the Queue for delivery as a Meeting request.

>    Attributes:
>    calendar
>    >    the Calendar name.

>    UID
>    >    the Event UID.

>    Body:
>    >    A set of one or more `To` elements specifying the recipient address(es).

calendarAccept
>    This operation places a calendaring item into a Calendar and sends a positive reply the item organizer. The existing items(s) with the same UID are replaced.

>    Attributes:
>    calendar
>    >    the Calendar name.

>    PARTSTAT

the acceptance status: `ACCEPTED`, `TENTATIVE`, `IN-PROCESS` (Tasks only), `COMPLETED` (Tasks only).

sendReply
  if this optional attribute is `no`, no reply message is send to the item organizer.

Body:

- The [iCalendar](#) element to place into the selected Calendar;
- optional [MIME](#) and/or [copyMIME](#) elements to add (the same as elements used with the `messageAppend` operation);
- optional `replyComment` element containing a text body - a comment to include into the reply message.

If the item is placed successfully, the iCalendar reply message is sent to the item organizer. Replies are not sent for Event items specifying the current user as an attendee with `RSVP=FALSE` parameter, or if the request `sendReply` attribute is set to `no`.

calendarDecline
  This operation rejects a calendaring item and sends a negative reply the item organizer.

  Attributes:
calendar
  the Calendar name. This parameter is optional.
  If specified, the items with the same UID are removed from this Calendar.
  If the item has the `recurrenceId` element, then only this exception is removed from the Calendar.

sendReply
  if this optional attribute is `no`, no reply message is send to the item organizer.

Body:

- The [iCalendar](#) element to decline.
- optional `replyComment` element containing a text body - a comment to include into the reply message.

Use this operation when a user decides not to attend a meeting stored as a published item in a Calendar. (if the current user is the meeting organizer, use the `calendarCancel` operation instead).
Specify the published item itself and the `calendar` attribute to remove the item from the Calendar.
The operation sends a negative reply message to the item organizer (unless the `sendReply` attribute is set to `no`).

Use this operation when a user opens a request item (in an incoming E-mail) and decides to decline that request.
Specify the request item, and do not specify any `calendar` attribute.
The operation sends a negative reply message to the item organizer (unless the `sendReply` attribute is set to `no`).

Use this operation when a user opens a cancellation request item (in an incoming E-mail), and decides to remove the canceled item from the Calendar.
Specify the cancellation request item and the `calendar` attribute (usually - the Main Calendar name), to remove the corresponding item(s) from that calendar.
The operation does not send any reply message.

If the specified item is a request, the operation sends a negative reply message to the item organizer.
If the specified item is a cancel request item, the operation does not send a reply message to the item organizer; specify the `calendar` attribute to remove the corresponding item from the calendar.

calendarImport
  This operation parses an uploaded file, which should contain [iCalendar](#)/[vCalendar](#) items. The resulting items are copied into a Calendar. The existing items(s) with the same UID are removed.

  Attributes:
calendar

the Calendar name.

uploadID
a string identifying a file in the "uploaded file set".

## freeBusyRead

This operation retrieves the FreeBusy information for the specified Account.

Attributes:

userName
the target Account name. If this attribute is not specified, the current Account FreeBusy information is retrieved.

timeFrom, timeTill
the beginning and the end of the time interval (time values should be specified for the selected time zone).

The Server sends a `freeBusyData` data message.

The Server sends the following data messages:

## events

This synchronous data message is sent when the Server processes the `findEvents` request.

Attributes:

calendar, timeFrom, timeTill, skip
the same as in the `findEvents` request.

items
the total number of Events found.

Body:
a set of `event` elements for each Event found.

Attributes:

UID
the Event message UID (a number).

itemUID
the Event item (iCalendar) UID string.

timeFrom
the time when this Event starts (in the selected time zone). This attribute is not included for "all-day" Events when the `byAlarm` mode is not set.

dateFrom
the date when this Event starts (in the selected time zone). This attribute is included for "all-day" Events only, and only when the `byAlarm` mode is not set.

duration
the Event duration (in seconds).

alarmTime
this attribute is included only if the request contained the `byAlarm` attribute. The attribute value specifies the time (in the selected time zone) when the Event Alarm should take place.

busyStatus
the Event busy status (BUSY, TENTATIVE, UNAVAILABLE, FREE).

organizer
this attribute is present and it has the `yes` value if the current user is the organizer of the Event.

recurrence

this attribute is present and it has the <code>yes</code> value if the Event is a recurrent one.

recurrenceId
> this attribute is present if the found Event is an exception of its main Event. The attribute value is a time stamp identifying this exception.

attendees
> this attribute is present if the Event has attendees. The attribute value is the number of attendees.

location
> this attribute is present if the Event has the Location element. The attribute value is the Location element value, optionally shortened.

priority
> this attribute is present if the Event has its priority set.

Body:
> Event fields: <code>summary</code>

## tasks

This synchronous data message is sent when the Server processes the <u>findTasks</u> request.

Attributes:

<code>calendar, timeFrom, timeTill, skip</code>
> the same as in the <u>findTasks</u> request.

items
> the total number of tasks found.

Body:
> a set of <code>task</code> elements for each Task found.

Attributes:

UID
> the Task message UID (a number).

itemUID
> the Task item (iCalendar) UID string.

timeFrom
> the time when this Task starts (in the selected time zone).

due
> the time this Task is due (in the selected time zone).

percent-complete
> the Task completion stage; a number, usually in the 0..100 range.

alarmTime
> this attribute is included only if the request contained the <code>byAlarm</code> attribute. The attribute value specifies the time (in the selected time zone) when the Task Alarm should take place.

organizer
> this attribute is present and it has the <code>yes</code> value if the current user is the organizer of the Task.

recurrence
> this attribute is present and it has the <code>yes</code> value if the Task is a recurrent one.

recurrenceId
> this attribute is present if the found Task is an exception of its main Task. The attribute value is a time stamp

identifying this exception.

attendees
this attribute is present if the Task has assignees. The attribute value is the number of assignees.

location
this attribute is present if the Task has the Location element. The attribute value is the Location element value, optionally shortened.

priority
this attribute is present if the Task has its priority set.

Body:
Task fields: summary

## calendarReport
These asynchronous data messages are sent when the Calendar data is modified.

Attributes:
calendar
the Calendar name.

mode
this optional attribute specifies the type of notification.

- If the attribute value is notify some calendar messages have been added, modified, or deleted. The client is expected to re-send the findEvents request for this Calendar.

After the Server sends a "notify-mode" calendarReport message to the Client, the Server may choose not to send further "notify-mode" messages for this Calendar until the client performs the findEvents operation on this Calendar.

## calendarMessage
This synchronous data message is sent when the Server processes the calendarRead request.
This message is the same as folderMessage, but instead of the folder attribute it contains the calendar attribute with the Calendar name.

## calendarItem
This synchronous data message is sent when the Server processes the calendarReadItem request.

Attributes:
calendar, UID
the copy of the calendarReadItem request attributes.

Body:
the xml presentation of the calendar item (vevent, vtodo).

## freeBusyData
This synchronous data message is sent when the Server processes the freeBusyRead request.

Attributes:
userName
the copy of the same freeBusyRead request attribute.

Body:

> the <u>vfreebusy</u> object containing a set of `freebusy` elements. All time values returned are specified as the local time in the Time Zone selected for the current user.

> After the Server sends a "notify-mode" `calendarReport` message to the Client, the Server may choose not to send further "notify-mode" messages for this Calendar until the client performs the `findEvents` operation on this Calendar.

---

# Signaling

A Client should use the "bind" operation to start receiving signals directed to the authenticated user.

A Client can maintain one or several concurrent communication sessions ("calls"). Each call is identified with its `callLeg` identifier. The `callLeg` attribute is present in all call-related operation requests and in all Server data messages related to that call. Each call is independent.

For each call, a Client should be able to create one or several "media objects", which implement actual media (audio, video, etc.) communications. The media descrptor element ("SDP elements") are data elements (text or XML) that a Client retrieves from and sends to "media objects".
As a minimum, it should be possible to perform the following operations on each media object:

- retrieve an "offer" SDP from the media object, and then send it an "answer" SDP that describes the media object of the communication peer; or
- send an "offer" SDP describing a communication peer to the media object, and then retrieve an "answer" SDP from that media object.

If it is not possible to re-send an "offer" SDP to an already active media object, a new media object should be created, and the new "offer" SDP is sent to. If the "answer" SDP is successfully retrieved from the new media object, the old media object should be disposed of.

If it is not possible to re-retreive an "offer" SDP from an already active media object, a new media object should be created, and an "offer" SDP is retreieved from it. When the "answer" SDP received and sent to the new media object, the old media object should be disposed of.

A media object supports "forking" if it is possible to retrieve a single "offer" SDP from that media object, and then send several "answer" SDP elements to that media object, establishing several concurrent media communication channels.

Many messages and operation requests described in this section may contain an "SDP descriptor" as the XML body.
The SDP descriptor may be specified using the <u>XML presentation</u>, or as an `sdpText` XML element with the SDP descriptor in the native SDP text format.
The <u>signalBind</u> operation request specifies the format that the Server will be using in its messages (see below).

signalBind
> This operation allows the current XIMSS session to receive signals directed to the authenticated user.

> Attributes:
> clientID
> > an optional parameter specifying a name for this session. If not specified, the Server generates a unique one.

> mode
> > if this attribute exists, and its value is `fixed`, the Server rejects the operation if the Account already has an open session using the same name;
> > if this attribute exists, and its value is `kill`, the Server closes the currently opened Account session with the same name (if any).
> > if this attribute does not exist or it has some other value, the Server generates a unique name for this session if the Account already has an open session that uses the specified name;

presence
>    if this optional attribute exists, and its value is `yes`, the client starts to receive [Roster and Presence](#) notifications.

readIM
>    if this optional attribute exists, and its value is `1`, the `readIM` messages use the "extended" format.

>    Body (optional):
>>    the XML presentation of the client [SDP](#) descriptor.
>>    The descriptor is used to specify the client capabilities (ability to accept audio/video calls) and to detect "far-end NAT" configurations.
>>    **Note:** to detect "far-end NAT" configurations, the SDP descriptor must contain the `ip` attribute with the default media IP address used by the client (see [below](#)).
>>    An `sdpText` XML element can be used instead of the [SDP](#) element. The `sdpText` XML element body should be text containing SDP data in the SDP native format.
>>    If the `sdpText` is used, then SDP information in `callIncoming`, `callProvisioned`, `callConnected`, and `callUpdated` Server messages is also presented using the `sdpText` XML element.

signalUnbind
>    After this operation is complete, the current XIMSS session does not receive signals directed to the authenticated user.

callKill
>    Use this operation to end the call and release all associated resources.
>    If the call was in the 'calling' state, the outgoing call is canceled.
>    If the call was in the 'alerting' state, the incoming call is rejected.
>    If the call was in the 'connected' state, the disconnect signal is sent to the peer.

>    Attributes:
>    callLeg
>>        the call identifier.

>    After this operation succeeds, it is possible to create a new call with the same identifier.

Outgoing calls are initiated by the Client using the [callStart](#) operation request. The Client should generate a unique `callLeg` attribute value for the `callStart` request. All other requests and messages related to this call will have the same `callLeg` attribute value.
When an outgoing call is started, the Server may send:

- zero, one, or several [callProvisioned](#) messages
- zero, one, or several [callUpdateRequest](#) messages
- zero or one [callUpdateSolicited](#) message

Each of these messages contains a `tag` attribute, identifying an "early dialog" (a "ringing" device) .
When an outgoing call is started, the Client may send:

- zero, one, or several [callUpdate](#) operation requests for "early" dialogs established with [callProvisioned](#), [callUpdateRequest](#), [callUpdateSolicited](#) messages. For each [callUpdate](#) operation requests, the Server sends back a [callUpdated](#) message.

An outgoing call becomes an "established" call when the Server sends a [callConnected](#) message.
An outgoing call fails if the Server sends a [callDisconnected](#) message.
The Client can cancel a pending outgoing call using the [callKill](#) operation.

Incoming calls are initiated by the Server sending to the Client a [callIncoming](#) message.
The Server generates a unique `callLeg` value for the `callIncoming` message. All other requests and messages related to this call will have the same `callLeg` attribute value. The Server-generated `callLeg` value for incoming calls use the `inp` prefix, so the Client should not use this prefix for the `callLeg` values it generates for outgoing calls.
When an incoming call is received, the Server may send:

- zero, one, or several `callUpdateRequest` messages

When an incoming call is received, the Client may send:

- zero, one, or several `callProvision` operation requests. For each `callProvision` operation request, the Server sends back a `callUpdated` message.
- zero, one, or several `callUpdate` operation requests. For each `callUpdate` operation request, the Server sends back a `callUpdated` message.

An incoming call fails (it is cancelled) if the Server sends a `callDisconnected` message.
The Client can end processing an incoming call by rejecting the call with the `callReject` operation, or by redirecting the call with the `callRedirect` operation.
The Client can accept an incoming call with the `callAccept` operation, making it an "established" call.

When an outgoing call connects or when an incoming call is accepted, the call becomes an "established" one. Media in an established call flows in both directions (unless one party decides to put the call "on hold").
When a call is established, the Server may send:

- zero, one, or several `callUpdateRequest` messages
- zero, one, or several `callUpdateSolicited` messages

When a call is established, the Client may send:

- zero, one, or several `callUpdate` operation requests. For each `callUpdate` operation request, the Server sends back a `callUpdated` message.

An established call ends if the Server sends a `callDisconnected` message.
The Client can end an established call using the `callKill` operation.

`callStart`
       Use this operation to start an outgoing call.

       Attributes:
       `callLeg`
            the new call identifier. The current XIMSS session should have no other call with this identifier.

       `peer`
            an E-mail address or a SIP URI to call.

       `peerName`
            an optional string - the real name of the callee.

       Body (optional):
            An [SDP descriptor](#).

       If the `callStart` request succeeds, the new call object is created and an outgoing call is initiated.
       **Note:** the client should be ready to process call-related Server messages even before the client receives the positive response for this `callStart` request.
       **Note:** if the call fails (the `callDisconnected` message is received), the client still needs to use the `callKill` operation to release the resources associated with the call, and to allow itself to reuse the call identifier.

       **Media Handling**

       Call with SDP: the Client creates a media object (called "untagged" object).
       The Client should configure the created media object to operate in the "receiving-only" mode.
       The Client retrieves an "offer SDP" from that untagged media object, and sends that SDP with the `callStart` request.
       **Note:** to process "behind-the-NAT" calls correctly, the SDP descriptor must contain the `ip` attribute with the default media

IP address used with the client.

Call without SDP: the Client does not create any media object. The Cleint sends the `callStart` request without an SDP body element.

Optionally, the Client should be prepared to create additional media objects and to build an "association mapping", where each created media object is accociated with a string - a value of the `tag` attribute of the receieved <u>callProvisioned</u> and/or <u>callUpdateRequest</u> messages.

`callProvision`
>Use this operation to provision an incoming call (the call object is created when the <u>callIncoming</u> message is received from the Server). Call provisioning informes the caller that the callee is being alerted (the callee's "phone rings").
>
>Attributes:
>`callLeg`
>>the incoming call identifier.
>
>Body (optional):
>>An <u>SDP descriptor</u>.
>
>When the operation completes, the Client will receive the <u>callUpdated</u> message from the Server.
>
>**Media Handling**
>
>If the Client wishes to send "early media" to the caller, the Client sends a `callProvision` request with SDP. "Early media" *may* be played by the caller's device instead of its default "ringback" tones.
>To send early media, the Client should create an untagged media object.
>The Client should configure the created media object to operate in the "sending-only" mode.
>
>If the <u>callIncoming</u> message contained an SDP:
>>That SDP element is an "offer".
>>The Client should send this "offer" SDP to the untagged media object.
>>Then the Client should retrieve the "answer" SDP from that media object and send it to the Server using the `callProvision` request.
>>Then the Client can instruct the media object to play the desired "early media".
>
>If the <u>callIncoming</u> message did not contain an SDP:
>>The Client should retrieve an "offer" SDP from the untagged media object and send it to the Server using the `callProvision` request.
>>When the Client receives the <u>callUpdated</u> message from the Server, it will contain an error code or an "answer" SDP.
>>If the "answer" SDP is received, the Client should send it to the untagged media object, and then the Client can instruct the media object to play the desired "early media".

`callRedirect`
>Use this operation to redirect an incoming call.
>
>Attributes:
>`callLeg`
>>the incoming call identifier.
>
>`fork`
>>if this optional attribute exists, and its value is `yes`, the call is directed to the specified destinations, but it is still pending for this session, and it can be accepted or rejected.
>
>Body:

one or more XML `To` elements. Each element should have a text body specifying a URI to redirect the call to.

**Note:** the client still needs to use the `callKill` operation to release the resources associated with the call and to allow itself to reuse the call identifier.

Example:
Redirecting an incoming call `inp003` to the `user1@example.com` and `user2@example.com`.

```
C:<callRedirect id="A018" callLeg="inp003" >
    <To>user1@example.com</To><To>user2@example.com</To>
   </callRedirect>
S:<response id="A018"/>
```

### Media Handling

The Client should dispose of all media objects associated with this call.

`callReject`
Use this operation to reject an incoming call. You can also use the `callKill` operation, but this operation allows you to specify an error code.

Attributes:
`callLeg`
    the incoming call identifier.

`signalCode`
    a numeric code (defined with the SIP standards) to pass to the Signal component as an error code.
    Use the 486 code ("Busy here") to signal that this device is "busy" (but other devices registered for this user will continue to ring).
    Use a 6xx code (600 - "Busy Everywhere" or 603 - "Declined") to signal that the user does not want to accept this call on any device (all other devices will stop ringing, too).
    If this parameter is not specified, the 603 code is used.

**Note:** the client still needs to use the `callKill` operation to release the resources associated with the call and to allow itself to reuse the call identifier.

Example:
Rejecting an incoming call `inp004` with the 603 "Declined" code:

```
C:<callReject id="A020" callLeg="inp004" signalCode="603" />
S:<response id="A020"/>
```

### Media Handling

The Client should dispose of all media objects associated with this call.

`callAccept`
Use this operation to accept an incoming call.

Attributes:
`callLeg`
    the incoming call identifier.

Body (optional):
    An SDP descriptor.

When the operation completes, the Client will receive the `callUpdated` message from the Server.

### Media Handling

The Client should create an untagged media object, if it has not been created yet.

If the Client has already sent a `callProvision` request with SDP:
>    The `callAccept` request should contain no SDP element, and the `callUpdated` message will not contain any SDP element.

If the Client has not sent any `callProvision` request with SDP, and the `callIncoming` message contained an SDP:
>    That SDP element is an "offer".
>    The Client should send this "offer" SDP to the untagged media object.
>    Then the Client should retrieve the "answer" SDP from that media object and send it to the Server using the `callAccept` request.
>    When the connection is established, the Client will receive the `callUpdated` message without an SDP from the Server.

If the Client has not sent any `callProvision` request with SDP, and the `callIncoming` message did not contain an SDP:
>    The Client should retrieve an "offer" SDP from the untagged media object and send it to the Server using the `callAccept` request.
>    The Client will receive the `callUpdated` message from the Server, which contains an "answer" SDP.
>    The Client should send that "answer" SDP to the untagged media object.


The Client should configure the "untagged" media object to stop playing "early media" and to operate in the "sending and receiving" mode.

callUpdate
>    Use this operation to update a call (to modify the call SDP descriptor). The Client may need to use this function if it places a call on hold or resumes it, when it changes the number and/type of the media streams (for example, adds a vidio stream to an audio call), switches to a different media object, etc.

>    Attributes:
>    callLeg
>>        the call identifier.

>    tag
>>        optional: the "to-tag" (dialog/fork ID) of an "early" dialog; should be used when updating an outgoing call.


>    Body (optional):
>>        An [SDP descriptor](#).


When the operation completes, the Client will receive a `callUpdated` message from the Server.
If the operation fails, the `callUpdated` message contains `signalCode` and `errorText` attributes. If the operation succeeds, the `callUpdated` message does not contain these attributes.

**Media Handling**

If the Client does not want to modify the call SDP descriptor, it should send the `callUpdate` request without an SDP descriptor. The Server will send back a `callUpdated`message without an SDP element.

To modify the call SDP descriptor, the Client should retrieve a new "offer" SDP from the untagged media object. Alternatively, it should create a new media object and retrieve the "offer" SDP from it. This "offer" SDP should be sent using the `callUpdate` request.
If the `callUpdated` message received does not contain an `errorText` attribute, it contains an "answer" SDP, and the Client should send it to the same media object. If it is a newly created media object, the old "untagged" media object should be removed, and the new one should become the current "untagged" media object.
If the operation fails, the `callUpdated` message contains an `errorText` attribute. In this case, the untagged media object should be reverted to its previous state. If a new media object has been created to retrieve an "offer" SDP, that new media

object should be removed.

## callTransfer
Use this operation to transfer the connected peer to a different party.

Attributes:
callLeg
the call identifier.

peer
an E-mail address or a SIP URI to transfer the call to. Use this attribute to initiate a "blind transfer" operation.

otherLeg
the call identifier. It should specify some other "call" in this session. Both calls should be in the 'connected' state. Use this attribute to complete an "attended transfer" operation. The remote peer of the "callLeg" call is connected to the remote peer of the "otherLeg" call.

If the otherLeg attribute is specified, the peer attribute is ignored.

If the call transfer operation succeeds, the "callLeg" call is disconnected (the callDisconnected data message is sent to the client). Additionally, in the case of an "attended transfer", the client receives the callDisconnected message for the "otherLeg" call, too.

If the call transfer operation fails, a callOpFailed data message is sent to the client.

## callUpdateAccept
Use this operation to accept call SDP modifications.

Attributes:
callLeg
the call identifier.

Body (optional):
An SDP descriptor.

The Client should send this request in response to callUpdateRequest, callProvisioned, and callConnected messages. See the description of these messages for the details.

## callUpdateReject
Use this operation to reject call SDP modifications.

Attributes:
callLeg
the call identifier.

signalCode
the numeric code (defined with the SIP standards) to pass to the Signal component as an error code.
If this attribute is absent, the 488 code ("Not Acceptable Here") is used.

Body:
none.

The Client should send this request instead of the callUpdateAccept request if the Client fails to perform the requested SDP modification.
See callUpdateAccept description for the details.

## callSendDTMF

Use this operation to send a DTMF signal via the signaling path.

Attributes:
`callLeg`
the call identifier.

Body:
a one-symbol string with the DTMF code to send (10 for '*', 11 for '#').

If the operation succeeds, the server sends a `callOpCompleted` message.

If the operation fails, the server sends a `callOpFailed` message.

`callSendInfo`
Use this operation to send an INFO signal to the connected peer.

Attributes:
`callLeg`
the call identifier.

Body:
a `MIME` XML element:

Attributes:
`type`
INFO content Content-Type.

`subtype`
INFO content subtype (optional).

Body:
a string with INFO content data.

If the operation succeeds, the server sends a `callOpCompleted` data message.

If the operation fails, the server sends a `callOpFailed` data message.

`makeCall`
Use this operation to establish a call using any registered device or client. The Server initiates a call to the authenticated Account ("self-call"), causing all its registered devices and clients to ring. When any device answers the call, that device is instructed to call the specified address (or telephone number).

Attributes:
`peer`
an E-mail address or a SIP URI to call.

`callerParams`
an optional attribute containing SIP R-URI (request-URI) parameters for the "self-call".

The operation completes as soon as the CG/PL task implementing this functionality is started. Asynchronous `makeCallReport` data messages are sent when the status of this task changes. When the call attempt completes, an empty `makeCallReport` data message is sent.

The Server can send the following data messages:

`callDisconnected`
>These asynchronous data messages are sent when an outgoing call fails, when an incoming call is canceled, or when an established call disconnects:
>
>Attributes:
>`callLeg`
>>the call identifier.
>
>`signalCode`
>>this optional attribute specifies the numeric signaling error code.
>
>`errorText`
>>optional: specifies the call disconnect reason.
>
>**Note:** the client still needs to use the `callKill` operation to release the resources associated with the call and to allow itself to reuse the call identifier.
>
>**Media Handling**
>
>Dispose of all media objects associated with this call.

`callProvisioned`
>These asynchronous data messages are sent when there is an outgoing call in progress, after a <u>callStart</u> request was sent to the Server:
>
>Attributes:
>`callLeg`
>>the call identifier.
>
>`callId`
>>the call Call-Id string
>
>`tag`
>>the "to-tag" (dialog/fork ID) of an outgoing "early" dialog.
>
>Body (optional):
>>An <u>SDP descriptor</u>.
>
>In response, the Client should send a <u>callUpdateAccept</u> or a <u>callUpdateReject</u> request to the Server.
>
>**Media Handling**
>
>If the `callProvisioned` message does not contain an SDP element:
>>if there is a media object or a ringback player associated with the `tag` attribute value, the Client should ignore this message.
>>Otherwise, the Client should create a ringback player (which plays a "ringback" sound, informing the user that the callee is being alterted), and associate this player with the `tag` attribute value.
>
>If the `callProvisioned` message contains an SDP element, and there is a media object associated with the `tag` attribute value:
>>The SDP element is an "offer".
>>The Client should use this "offer" SDP to update the associated media object. Then the Client should retrieve the "answer" SDP from that media object and send it to the Server using the <u>callUpdateAccept</u> request.
>
>If the `callProvisioned` message contains an SDP element, there is no media object associated with the `tag` attribute value, and the <u>callStart</u> request contained an "offer" SDP:

The SDP element is an "answer".

The Client should remember this "answer" SDP associated with the `tag` attribute value, replacing any older SDP associated with this `tag` attribute value.

If the "untagged" media object supports forking, the Client should send this "answer" SDP to that media object; the Client should also remove any ringback object associated with the `tag` attribute value.

If the "untagged" media object does not support "forking", the Client should create a ringback player and associate it with the `tag` attribute value, if a ringback player associate with this value does not already exist.

If the `callProvisioned` message contains an SDP element, there is no media object associated with the `tag` attribute value, and the `callStart` request did not contain an SDP:

The SDP element is an "offer".

The Client should create a new media object and associate it with the `tag` attribute value. If there is a ringback object associated with the `tag` attribute value, the Client should remove it.

The Client should send this "offer" SDP to the newly created media object, retrieve the "answer" SDP from the media object, and send this "answer" SDP to the Server using the `callUpdateAccept` request.

If the Client cannot process the `callProvisioned` message, it should send the `callUpdateReject` request to the Server.

## callConnected

These asynchronous data messages are sent when an outgoing call (initiated with a `callStart` request) succeeds, and the call is established:

Attributes:

callLeg
the call identifier.

callId
the call Call-Id string

peer
the address of the peer which has accepted this call.

tag
the "to-tag" (dialog/fork ID) of the established dialog.

Body (optional):
An SDP descriptor.

In response, the Client should send a `callUpdateAccept` or a `callUpdateReject` request to the Server.

**Media Handling**

If there is a media object associated with the `tag` attribute value, and the `callConnected` message does not contain an SDP element:

The Client should remove all other media objects and all ringback players. The selected media object becomes the "untagged" media object.

If there is a media object associated with the `tag` attribute value, and the `callConnected` message contains an SDP element:

The SDP element is an "offer".

The Client should send this "offer" SDP to the selected media object. Then the Client should retrieve the "answer" SDP from that media object and send it to the Server using the `callUpdateAccept` request.

The Client should remove all other media objects and all ringback players. The selected media object becomes the "untagged" media object.

If there is no a media object associated with the `tag` attribute value, the `callStart` request contained an "offer" SDP, and the `callConnected` message contains an SDP element:

The SDP element is an "answer".

The Client should send this "answer" SDP to the "untagged" media object. If the media object supports "forking" and

some "answer" SDP have been sent to it, the Client should instruct the media object so stop these other media communications.

The Client should remove all tagged media objects and ringback players.

If there is no a media object associated with the `tag` attribute value, the <u>callStart</u> request contained an "offer" SDP, and the `callConnected` message does not contain an SDP element:

The SDP element is an "answer".

If the media object supports "forking", then some "answer" SDP associated with the `tag` attribute value has been already sent to it. The Client should instruct the media object to stop all other media communications.

If the media object does not support "forking", then some "answer" SDP received with a <u>callProvisioned</u> message and associated with the `tag` attribute value should be "remembered". The Client should send this remembered "answer" SDP to the untagged media object.

The Client should remove all tagged media objects and ringback players.

If there is no a media object associated with the `tag` attribute value, the <u>callStart</u> request did not contain an "offer" SDP:

The `callConnected` message must contain an SDP element, and that SDP element is an "offer".

The Client should create a new media object, and to send this "offer" SDP to the new media object. Then the Client should retrieve the "answer" SDP from that media object and send it to the Server using the <u>callUpdateAccept</u> request.

The Client should remove all other media objects and all ringback players. The created media object becomes the "untagged" media object.

The "untagged" media object (the only media object left) should be instructed to use the "sending and receiving" mode.

callIncoming

These asynchronous data messages are sent when new incoming calls are received (you need to use the `signalBind` operation to receive incoming calls):

Attributes:
callLeg
the call identifier. The Server generates these identifiers itself.

peer
the remote peer E-mail address.

peerName
optional; the remote peer real name.

callId
the call Call-Id string

transferredBy
optional; the E-mail address of the person who has transferred this call.

Body (optional):
An [SDP descriptor](#).

**Media Handling**

The Client may create an "untagged" media object immediately, or it may create it when the Client sends <u>callProvision</u> or <u>callAccept</u> requests.

If the `callIncoming` message contains an SDP element:
The SDP element is an "offer".
The Client should pass this SDP to the "untagged" media object when it is created.

callUpdateRequest

These asynchronous data messages are sent when the communication peer wants to update the call parameters (such as SDP).

Attributes:
callLeg
>    the call identifier.

callId
>    the call Call-Id string

tag
>    optional: the "to-tag" (dialog/fork ID) of an "early" dialog.

Body (optional):
>    An SDP descriptor.

The client should answer by sending the callUpdateAccept or callUpdateReject request to the Server.

**Media Handling**

If the callUpdateRequest message does not contain an SDP element, the Client can ignore it.
If the callUpdateRequest message contains an SDP element:
>    This element is an "offer" SDP.
>    If the call is still outgoing, the Client should remove any "ringback player" associated with this tag.
>    The Client should find the media object associated with the tag attribute (or create a new media object if none is found).
>    If no tag attribute is specified, the "untagged" media object should be used.
>    The "offer" SDP from the callUpdateRequest message should be used to update the selected media object.
>    The client should retrieve the "answer" SDP from the selected media object, and send it to the Server using the callUpdateAccept request.
>    If the client does not want to or cannot accept the new "offer" SDP, it should inform the Server using the callUpdateReject request.

callUpdated
>    These asynchronous data messages are sent when an existing call is updated (placed on hold, switched to a different remote peer, etc.).

Attributes:
callLeg
>    the call identifier.

peer
>    optional; the remote peer E-mail address. It is sent when a call has been transferred.

peerName
>    optional; the remote peer real name.

signalCode
>    this optional attribute specifies the numeric signaling error code.

errorText
>    optional; the error message.

Body (optional):
>    An SDP descriptor.

This message is received in response to the callUpdate, callProvision, callAccept requests. See the description of those

requests for the details.

If the operation fails, the `signalCode` and `errorText` attributes are present.

If the operation succeeds, the `signalCode` and `errorText` attributes are absent, and an [SDP descriptor](#) *may* be present.

callUpdateSolicited

These asynchronous data messages are sent when the communication peer wants to update the call SDP descriptor, but it wants this Client to provide an SDP offer.

Attributes:
callLeg
the call identifier.

tag
optional: the "to-tag" (dialog/fork ID) of an "early" dialog.

Body (optional):
An [SDP descriptor](#).

The Server sends these messages only when a call is established or when an outgoing call is pending.

**Media Handling**

The Client should execute the [callUpdate](#) operation with a new "offer" SDP.

If the Client cannot initiate a `callUpdate` operation (for example, it cannot create a new media object), it should send the [callUpdateReject](#) request to the Server.

For a pending outgoing call, the Client should create a "tagged" media object, from which it should retrieve the "offer" SDP for the [callUpdate](#) request.

callOpCompleted

These asynchronous data messages are sent when in-call operations (such as `callUpdate`, `callSendDTMF`, `callSendInfo`) are completed.

Attributes:
callLeg
the call identifier.

callOpFailed

These asynchronous data messages are sent when in-call operations (such as `callUpdate`, `callSendDTMF`, `callSendInfo`) fail.

Attributes:
callLeg
the call identifier.

signalCode
the numeric signaling error code.

errorText
the error message.

callDTMF

These asynchronous data messages are sent when a DTMF signal is received via the signaling path.

Attributes:
callLeg

the call identifier.

Body:

A string with the numeric DTMF code received.

## callTransferred

These asynchronous data messages are sent when the remote peer has transferred the call to a different peer.

Attributes:
callLeg

the call identifier.

peer

the new peer address.

peerName

optional; the new peer real name.

callId

the call Call-Id string; call transfer usually results in a Call-Id change.

Body:

Empty.

## callInfo

These asynchronous data messages are sent when an INFO signal is received:

Attributes:
callLeg

the call identifier.

Body:

the same as in the callSendInfo request.

## makeCallReport

These asynchronous data messages are sent after the makeCall operation is initiated.

Body:

A string containing the current operation status. A message contains an empty body if the calling process completes.

Example Incoming-01. Incoming call session with an initial SDP. The Client does not send any custom "ringback" media to the caller.

```
The Client receives a callIncoming message with a Server-generated callLeg attribute.
S:<callIncoming id="A001" callLeg="inp01" peer="user@domain" >SDP:X(offer)</callIncoming>

The Client starts to alert the user (the Client starts "ringing").
The Client notifiers the caller that the user is being alerted:
C:<callProvision id="A001" callLeg="inp01"/>
S:<response id="A001"/>

The user instructs the Client to accept this incoming call.
The Client creates an "untagged" media object A, and sends the received SDP:X(offer) (an "offer" SDP) to that media object
A.
The Client retrieves an "answer" SDP from the media object A, and sends it to the Server, accepting the call:
C:<callAccept id="A002" callLeg="inp01">SDP:A(answer)</callAccept>
S:<response id="A002"/>
The Server sends back the callUpdated message:
S:<callUpdated callLeg="inp01"/>
```

```
Note: the callUpdated may arrive before arrival of the response message for the callAccept request.

The Client stops "ringing".
The call has connected, the media object A is instructed to work in the "send and receive" mode, communications started.

S:<callDisconnected callLeg="inp01" />
The call ends, the Client removes the "untagged" media object A.

The Client releases the Server resources associated with this call:
C:<callKill id="A003" callLeg="inp01"/>
S:<response id="A003"/>
```

Example Incoming-02. Incoming call session without an initial SDP. The Client does not send any custom "ringback" media to the caller.

```
The Client receives a callIncoming message with a Server-generated callLeg attribute.
S:<callIncoming id="A001" callLeg="inp01" peer="user@domain" />

The Client starts to alert the user (the Client starts "ringing").
The Client notifiers the caller that the user is being alerted:
C:<callProvision id="A001" callLeg="inp01"/>
S:<response id="A001"/>

The user instructs the Client to accept this incoming call.
The Client creates an "untagged" media object A, and retrieves an "offer" SDP from this media object: SDP:A(offer).
The Client sends this SDP to the Server, accepting the call:
C:<callAccept id="A002" callLeg="inp01">SDP:A(offer)</callAccept>
S:<response id="A002"/>
The Server sends back the callUpdated message with an "answer" SDP:
S:<callUpdated callLeg="inp01">SDP:X(answer)</callUpdated>
Note: the callUpdated may arrive before arrival of the response message for the callAccept request.

The Client stops "ringing".
The Client sends the received "answer" SDP to the media object A.
The call has connected, the media object A is instructed to work in the "send and receive" mode, communications started.

S:<callDisconnected callLeg="inp01" />
The call ends, the Client removes the "untagged" media object A.

The Client releases the Server resources associated with this call:
C:<callKill id="A003" callLeg="inp01"/>
S:<response id="A003"/>
```

Example Incoming-03. Incoming call session with an initial SDP. The Client does not send any custom "ringback" media to the caller. The call has been cancelled by the caller (or some other client or device has answered this call).

```
The Client receives a callIncoming message with a Server-generated callLeg attribute.
S:<callIncoming id="A001" callLeg="inp01" peer="user@domain" >SDP:X(offer)</callIncoming>

The Client starts to alert the user (the Client starts "ringing").
The Client notifiers the caller that the user is being alerted:
C:<callProvision id="A001" callLeg="inp01"/>
S:<response id="A001"/>

The incoming call is cancelled:
S:<callDisconnected callLeg="inp01" />
The call ends (it is a "missed" call for this Client), the Client stops "ringing".

The Client releases the Server resources associated with this call:
C:<callKill id="A002" callLeg="inp01"/>
S:<response id="A002"/>
```

Example Incoming-04. Incoming call session with an initial SDP. The Client does not send any custom "ringback" media to the caller. The call has been declined by the user.

```
The Client receives a callIncoming message with a Server-generated callLeg attribute.
S:<callIncoming id="A001" callLeg="inp01" peer="user@domain" >SDP:X(offer)</callIncoming>

The Client starts to alert the user (the Client starts "ringing").
The Client notifiers the caller that the user is being alerted:
C:<callProvision id="A001" callLeg="inp01"/>
S:<response id="A001"/>

The incoming call is declined:
S:<callReject id="A002" callLeg="inp01" signalCode="603" />
S:<response id="A002"/>
The call ends, the Client stops "ringing".

The Client releases the Server resources associated with this call:
C:<callKill id="A003" callLeg="inp01"/>
S:<response id="A003"/>
```

Example Incoming-05. Incoming call session with an initial SDP. The Client does not send any custom "ringback" media to the caller. The user has redirected the call to the voicemail application.

```
The Client receives a callIncoming message with a Server-generated callLeg attribute.
S:<callIncoming id="A001" callLeg="inp01" peer="user@domain" >SDP:X(offer)</callIncoming>

The Client starts to alert the user (the Client starts "ringing").
The Client notifiers the caller that the user is being alerted:
C:<callProvision id="A001" callLeg="inp01"/>
```

```
S:<response id="A001"/>

The incoming call is redirected:
S:<callRedirect id="A002" callLeg="inp01"><To>#voicemail</To></callRedirect>
S:<response id="A002"/>
The call ends, the Client stops "ringing".

The Client releases the Server resources associated with this call:
C:<callKill id="A003" callLeg="inp01"/>
S:<response id="A003"/>
```

Example Outgoing-01. Outgoing call session with an initial SDP.

```
The Client creates a media object A and makes it the "untagged" media object for the new call.
The Client retrieves the "offer" SDP from that object: SDP:A(offer)
C:<callStart id="A001" callLeg="c1" peer="user@domain">SDP:A(offer)</callStart>
S:<response id="A001"/>

S:<callProvisioned callLeg="c1" tag="device1"/>
The Client creates a "ringback player" so the user knows that the communication peer is being alerted.
C:<callUpdateAccept id="A002" callLeg="c1"/>
S:<response id="A002"/>

S:<callProvisioned callLeg="c1" tag="device1"/>
The Client already has a "ringback player" for this device (tag), so it does not do any additional media operation.
C:<callUpdateAccept id="A003" callLeg="c1"/>
S:<response id="A003"/>

S:<callConnected callLeg="c1" tag="device1">SDP:X(answer)</callConnected>
The Client removes the "ringback player", and passes the received SDP:X(answer) media descriptor (it's an "answer" SDP) to
its "untagged" media object A. The call has connected, the media object A is instructed to work in the "send and receive"
mode, communications started.
C:<callUpdateAccept id="A004" callLeg="c1"/>
S:<response id="A004"/>

S:<callDisconnected callLeg="c1" />
The call ends, the Client removes the "untagged" media object A.

The Client releases the Server resources associated with this call:
C:<callKill id="A005" callLeg="c1"/>
S:<response id="A005"/>
```

Example Outgoing-02. Outgoing call session with an initial SDP, multiple callee devices. Media objects do not support "forking". The Client does not support advanced outgoing call messages.

```
The Client creates a media object A and makes it the "untagged" media object for the new call.
The Client retrieves the "offer" SDP from that object: SDP:A(offer)
C:<callStart id="A001" callLeg="c1" peer="user@domain">SDP:A(offer)</callStart>
S:<response id="A001"/>

S:<callProvisioned callLeg="c1" tag="device1"/>
The Client creates a "ringback player" so the user knows that the communication peer is being alerted.
C:<callUpdateAccept id="A002" callLeg="c1"/>
S:<response id="A002"/>

S:<callProvisioned callLeg="c1" tag="device2"/>
The Client creates one more "ringback player", so the user knows that the communication peer is being alerted. Both players
are playing.
C:<callUpdateAccept id="A003" callLeg="c1"/>
S:<response id="A003"/>

S:<callProvisioned callLeg="c1" tag="device1">SDP:X(answer1)</callProvisioned>
The Client stores the SDP:X(answer1) received, associated with the tag "device1". No changes to the media, both players are
playing.
C:<callUpdateAccept id="A004" callLeg="c1"/>
S:<response id="A004"/>

S:<callProvisioned callLeg="c1" tag="device1">SDP:X(answer2)</callProvisioned>
The Client stores the SDP:X(answer2) received, associated with the tag "device1". It replaces the SDP:X(answer1) received
for this tag earlier. No changes to the media, both players are playing.
C:<callUpdateAccept id="A005" callLeg="c1"/>
S:<response id="A005"/>

S:<callUpdateRequest callLeg="c1" tag="device3">SDP:Y(offer)</callUpdateRequest>
C:<callUpdateReject id="A006" callLeg="c1" errorText="not implemented" />
S:<response id="A006"/>

S:<callUpdateSolicited callLeg="c1" tag="device4" />
C:<callUpdateReject id="A006" callLeg="c1" errorText="not implemented" />
S:<response id="A006"/>

S:<callConnected callLeg="c1" tag="device1">SDP:X(answer3)</callConnected>
The Client removes all "ringback players", and passes the received SDP:X(answer3) media descriptor (it's an "answer" SDP) to
its "untagged" media object A.

Alternatively, the <callConnected/> message could come without any SDP in it:
S:<callConnected callLeg="c1" tag="device1"/>
The Client removes all "ringback players", and retrieves the SDP:X(answer2) media descriptor, as it was the last one it
stored for the "device1" tag. The Client passes the received SDP:X(answer2) media descriptor (it's an "answer" SDP) to its
"untagged" media object A.

The call has connected, the media object A is instructed to work in the "send and receive" mode, communications started.
C:<callUpdateAccept id="A006" callLeg="c1"/>
S:<response id="A006"/>

The user instructs the Client to end this call, the Client ends this call and releases the Server resources associated with
this call:
S:<callKill id="A007" callLeg="c1" />
S:<response id="A007"/>
```

```
The call ends, the Client removes the "untagged" media object A.
```

Example Outgoing-03. Outgoing call session with an initial SDP, multiple callee devices. Media objects do not support "forking", additional messages from the called devices.

```
The Client creates a media object A and makes it the "untagged" media object for the new call.
The Client retrieves the "offer" SDP from that object: SDP:A(offer)
C:<callStart id="A001" callLeg="c1" peer="user@domain">SDP:A(offer)</callStart>
S:<response id="A001"/>

S:<callProvisioned callLeg="c1" tag="device1"/>
The Client creates a "ringback player" so the user knows that the communication peer is being alerted.
C:<callUpdateAccept id="A002" callLeg="c1"/>
S:<response id="A002"/>

S:<callProvisioned callLeg="c1" tag="device2"/>
The Client creates one more "ringback player", so the user knows that the communication peer is being alerted. Both players
are playing.
C:<callUpdateAccept id="A003" callLeg="c1"/>
S:<response id="A003"/>

S:<callProvisioned callLeg="c1" tag="device1">SDP:X(answer1)</callProvisioned>
The Client stores the SDP:X(answer1) received, associated with the tag "device1". No changes to the media, both players are
playing.
C:<callUpdateAccept id="A004" callLeg="c1"/>
S:<response id="A004"/>

S:<callProvisioned callLeg="c1" tag="device1">SDP:X(answer2)</callProvisioned>
The Client stores the SDP:X(answer2) received, associated with the tag "device1". It replaces the SDP:X(answer1) received
for this tag earlier. No changes to the media, both players are playing.
C:<callUpdateAccept id="A005" callLeg="c1"/>
S:<response id="A005"/>

S:<callUpdateRequest callLeg="c1" tag="device2">SDP:Y(offer)</callUpdateRequest>
The Client removes the "ringback player" associated with the "device2" tag.
The Client creates a new "tagged" media object B, and associates it with the "device2" tag. The media object B is
configured to work in the "receiving only" mode.
The Client passes the received SDP:Y(offer) as an "offer" SDP to this media object B.
The Client retrieves the "answer" SDP from that media object: SDP:B(answer), and passes it to the Server:
C:<callUpdateAccept id="A006" callLeg="c1">SDP:B(answer)</callUpdateAccept>
S:<response id="A006"/>

S:<callConnected callLeg="c1" tag="device1">SDP:X(answer3)</callConnected>
The Client removes all "ringback players", removes the "tagged" media object B, and passes the received SDP:X(answer3) media
descriptor (it's an "answer" SDP) to its "untagged" media object A.

Alternatively, the <callConnected/> message could come without any SDP in it:
S:<callConnected callLeg="c1" tag="device1"/>
The Client removes all "ringback players", removes the "tagged" media object B, and retrieves the SDP:X(answer2) media
descriptor, as it was the last one it stored for the "device1" tag. The Client passes the received SDP:X(answer2) media
descriptor (it's an "answer" SDP) to its "untagged" media object A.

Alternatively, the <callConnected/> message could come without any SDP in it, and with a different tag:
S:<callConnected callLeg="c1" tag="device2"/>
The Client removes all "ringback players", removes the "untagged" media object A, and makes the "tagged" media object B
(associated with the "device2" tag) the new "untagged" media object.

The call has connected, the "untagged" media object (be it A or B) is instructed to work in the "send and receive" mode,
communications started.
C:<callUpdateAccept id="A007" callLeg="c1"/>
S:<response id="A007"/>

The user instructs the Client to end this call, the Client ends this call and releases the Server resources associated with
this call:
S:<callKill id="A008" callLeg="c1" />
S:<response id="A008"/>
The call ends, the Client removes the "untagged" media object A.
```

Example Outgoing-04. Outgoing call session with an initial SDP, multiple callee devices. Media objects do not support "forking", additional messages from the called devices.

```
The Client creates a media object A and makes it the "untagged" media object for the new call.
The Client retrieves the "offer" SDP from that object: SDP:A(offer)
C:<callStart id="A001" callLeg="c1" peer="user@domain">SDP:A(offer)</callStart>
S:<response id="A001"/>

S:<callProvisioned callLeg="c1" tag="device1"/>
The Client creates a "ringback player" so the user knows that the communication peer is being alerted.
C:<callUpdateAccept id="A002" callLeg="c1"/>
S:<response id="A002"/>

S:<callProvisioned callLeg="c1" tag="device2"/>
The Client creates one more "ringback player", so the user knows that the communication peer is being alerted. Both players
are playing.
C:<callUpdateAccept id="A003" callLeg="c1"/>
S:<response id="A003"/>

S:<callProvisioned callLeg="c1" tag="device1">SDP:X(answer1)</callProvisioned>
The Client stores the SDP:X(answer1) received, associated with the "device1" tag. No changes to the media, both players are
playing.
C:<callUpdateAccept id="A004" callLeg="c1"/>
S:<response id="A004"/>

S:<callProvisioned callLeg="c1" tag="device1">SDP:X(answer2)</callProvisioned>
The Client stores the SDP:X(answer2) received, associated with the "device1" tag. It replaces the SDP:X(answer1) received
for this tag earlier. No changes to the media, both players are playing.
C:<callUpdateAccept id="A005" callLeg="c1"/>
```

```
S:<response id="A005"/>

S:<callUpdateRequest callLeg="c1" tag="device2">SDP:Y(offer1)</callUpdateRequest>
The Client removes the "ringback player" associated with the "device2" tag.
The Client creates a new media object B, and associates it with the "device2" tag. The media object B is configured to work
in the "receiving only" mode.
The Client passes the received SDP:Y(offer1) as an "offer" SDP to this media object B.
The Client retrieves the "answer" SDP from that media object: SDP:B(answer), and passes it to the Server:
C:<callUpdateAccept id="A006" callLeg="c1">SDP:B(answer)</callUpdateAccept>
S:<response id="A006"/>

S:<callUpdateRequest callLeg="c1" tag="device2">SDP:Y(offer2)</callUpdateRequest>
The Client uses the received SDP to update the existing media object B associated with the "device2" tag.
If this is not possible, the Client creates a new media object C, sends it the received SDP:Y(offer2) as an "offer" SDP,
and retrieves from that media object the "answer" SDP:C(answer). If it succeeds, it removes the media object B and makes
the newly created media object C a "tagged" media object associated with the "device2" tag.
The new media object C is configured to work in the "receiving only" mode.
The Client and passes the retrieved "answer" SDP to the Server:
C:<callUpdateAccept id="A007" callLeg="c1">SDP:C(answer)</callUpdateAccept>
S:<response id="A007"/>

S:<callUpdateSolicited callLeg="c1" tag="device3"/>
The Client creates a new media object D, and associates it with the "device3" tag.
The new media object D is configured to work in the "receiving only" mode.
The Client retrieves the "offer" SDP - SDP:D(offer) - from this media object, and passes it to the Server, using the
callUpdate operation.
C:<callUpdate id="A008" callLeg="c1">SDP:D(offer)</callUpdate>
S:<response id="A008"/>

The Server then sends the callUpdated message:
S:<callUpdated callLeg="c1" tag="device3">SDP:Z(answer)</callUpdated>
The Client should pass the received SDP to the media object D as an "answer" SDP.

S:<callConnected callLeg="c1" tag="device1">SDP:X(answer3)</callConnected>
The Client removes all "ringback players", removes all "tagged" media object (C and D), and passes the received SDP:X-3
media descriptor (it's an "answer" SDP) to its "untagged" media object A.

Alternatively, the <callConnected/> message could come without any SDP in it:
S:<callConnected callLeg="c1" tag="device1"/>
The Client removes all "ringback players", removes all "tagged" media object (C and D), and retrieves the SDP:X(answer2)
media descriptor, as it was the last one it stored for the tag="device1". The Client passes the received SDP:X(answer2)
media descriptor (it's an "answer" SDP) to its "untagged" media object A.

Alternatively, the <callConnected/> message could come without any SDP in it, and with a different tag:
S:<callConnected callLeg="c1" tag="device2"/>
The Client removes all "ringback players", removes the "untagged" media object A and the tagged media object D, and makes
the "tagged" media object C (associated with the "device2" tag) the new "untagged" media object.

Alternatively, the <callConnected/> message could come without any SDP in it, and with a different tag:
S:<callConnected callLeg="c1" tag="device3"/>
The Client removes all "ringback players", removes the "untagged" media object A and the tagged media object C, and makes
the "tagged" media object D (associated with the "device3" tag) the new "untagged" media object.

The call has connected, the "untagged" media object (be it A, C, or D) is instructed to work in the "send and receive"
mode, communications started.
C:<callUpdateAccept id="A007" callLeg="c1"/>
S:<response id="A007"/>

The user instructs the Client to end this call, the Client ends this call and releases the Server resources associated with
this call:
S:<callKill id="A008" callLeg="c1" />
S:<response id="A008"/>
The call ends, the Client removes the "untagged" media object A.
```

Example Outgoing-11. Outgoing call session without an initial SDP.

```
The Client does not create any media object before a call.
C:<callStart id="A001" callLeg="c1" peer="user@domain" />
S:<response id="A001"/>

S:<callProvisioned callLeg="c1" tag="device1"/>
The Client creates a "ringback player" so the user knows that the communication peer is being alerted.
C:<callUpdateAccept id="A002" callLeg="c1"/>
S:<response id="A002"/>

S:<callProvisioned callLeg="c1" tag="device1"/>
The Client already has a "ringback player" for this device (tag), so it does not do any additional media operation.
C:<callUpdateAccept id="A003" callLeg="c1"/>
S:<response id="A003"/>

S:<callConnected callLeg="c1" tag="device1">SDP:X-1</callConnected>
The Client removes the "ringback player", creates an "untagged" media object A, and passes the received SDP:X-1 media
descriptor (it's an "offer" SDP) to its "untagged" media object A.
The Client retrieves the "answer" SDP (SDP:A-1) from the media object A, and sends it to the Server:
C:<callUpdateAccept id="A004" callLeg="c1">SDP:A-1</callUpdateAccept>
S:<response id="A004"/>
The call has connected, the media object A is instructed to work in the "send and receive" mode, communications started.

S:<callDisconnected callLeg="c1" />
The call ends, the Client removes the "untagged" media object A.

The Client releases the Server resources associated with this call:
C:<callKill id="A005" callLeg="c1"/>
S:<response id="A005"/>
```

Example Outgoing-12. Outgoing call session without an initial SDP, multiple callee devices.

```
The Client does not create any media object before a call.
C:<callStart id="A001" callLeg="c1" peer="user@domain" />
S:<response id="A001"/>
```

```
S:<callProvisioned callLeg="c1" tag="device1"/>
The Client creates a "ringback player" so the user knows that the communication peer is being alerted.
C:<callUpdateAccept id="A002" callLeg="c1"/>
S:<response id="A002"/>

S:<callProvisioned callLeg="c1" tag="device2"/>
The Client creates one more "ringback player", so the user knows that the communication peer is being alerted. Both players
are playing.
C:<callUpdateAccept id="A003" callLeg="c1"/>
S:<response id="A003"/>

S:<callProvisioned callLeg="c1" tag="device2">SDP:X-1</callProvisioned>
The Client removes the "ringback player" associated with the "device2" tag.
The Client creates a media object A, makeing it a "tagged" object associated with the "device2" tag.
The Client sends the received SDP:X-1 as an "offer" SDP to the media object A.
The Client retrieves an "answer" SDP from the media object A (SDP:A-1) and sends it to the Server.
C:<callUpdateAccept id="A004" callLeg="c1">SDP:A-1</callUpdateAccept>
S:<response id="A004"/>

S:<callConnected callLeg="c1" tag="device1">SDP:Y-1</callConnected>
The Client removes the "ringback player", and the media object A (tagged with the "device2" tag).
The Client creates a media object B, making it the "untagged" media object, and sends the received SDP:Y-1 media descriptor
(it's an "offer" SDP) to this "untagged" media object B.
The Client retrieves an "answer" SDP from the media object B (SDP:B-1) and sends it to the Server.
The call has connected, the media object B is instructed to work in the "send and receive" mode, communications started.
C:<callUpdateAccept id="A005" callLeg="c1">SDP:B-1</callUpdateAccept>
S:<response id="A005"/>

Alternatively, the <callConnected/> message could come without any SDP in it and with a different device tag:
S:<callConnected callLeg="c1" tag="device2"/>
The Client removes all "ringback players", and makes the media object A (associated with the "device2" tag) the "untagged"
media object.
The call has connected, the media object A is instructed to work in the "send and receive" mode, communications started.
C:<callUpdateAccept id="A005" callLeg="c1"/>
S:<response id="A005"/>

The user instructs the Client to end this call, the Client ends this call and releases the Server resources associated with
this call:
S:<callKill id="A006" callLeg="c1" />
S:<response id="A006"/>
The call ends, the Client removes the "untagged" media object A.
```

# Instant Messaging

The Instant Messaging (IM) model assumes that a client maintains a separate window for each "peer", where a "peer" is some other IM user taking part in an IM conversation.

sendIM
>     This operation sends an Instant Message to the specified user. The Server sends a XIMSS response without waiting till the Instant Message is actually delivered (or failed).
>     There is no explicit IM session opening operation. If the Server does not have an open IM session with the specified peer, a new session is created.
>
>     Attributes:
>     peer
>>          the E-mail address of the user to send this message to.
>
>     clientID
>>          optional; the name or ID of the peer device to send this message to.
>
>     type
>>          optional; the message XMPP-style type - chat, groupchat, etc.
>
>     iqid
>>          optional; the message identifier string.
>
>     Body:
>
>     - the Instant Message text (in the UTF-8 encoding) or
>     - the body XML element, with the Instant Message text as its body, and, optionally, other XML message elements. Or
>     - the composing XML element (this element should be sent when the user is preparing the message but has not sent it yet). Or
>     - the paused XML element (this element should be sent when the user has stopped preparing the message before sending it). Or
>     - the gone XML element (this element should be sent when the user has finished the conversation).

the `subject` XML element; its text body is the message subject string.

`closeIM`
A request to close all IM sessions with the specified user.
A client application should send this request when it closes an IM conversation window.

Attributes:
`peer`
the user E-mail address.

The Server sends the following data messages:

`readIM`
These asynchronous data messages are sent when the Server receives an Instant Message for the client user.

Attributes:
`peer`
the sender E-mail address.

`peerName`
optional; the sender real name.

`clientID`
optional; the name or ID of the peer device this message was sent from.

`subject`
optional; the message subject string (if the basic format is used).

`iqid`
optional; the message identifier string.

Body:

- the `composing` XML element; this element is sent when the remote peer is preparing (typing) the message but has not sent it yet. Or
- the `paused` XML element; this element is sent when the remote peer has stopped preparing the message. Or
- the `gone` XML element; this element is sent when the remote peer has finished the conversation. Or
- the message content in the basic or extended format, as requested with the last `signalBind` operation:

  simple format:
  the Instant Message text (in the UTF-8 encoding)

  extended format:
  the `body` XML element with the body containing the Instant Message text, and optional additional XML message elements.

  the `subject` XML element; its text body is the message subject string.

There is no explicit Instant Message session opening operation. If the client has no open conversation window for the specified user, it should open a new one.

`errorIM`
These asynchronous data messages are sent when the Server fails to deliver an Instant Message.

Attributes:

`peer`
> the recipient E-mail address.

`errorText`
> the error message text.

`signalCode`
> the numeric signaling error code.

`sendid`
> the `id` attribute of the `sendIM` operation that sent the failed Instant Message.

---

# Roster and Presence

The Roster and Presence model resembles that of the [XMPP](#) protocol.

A Roster is a set of items, each item describing a "contact" - some other local or remote user. The user can monitor the presence information of a "contact", and a "contact" can be granted a right to monitor the user's presence information.

`rosterList`
> This operation retrieves all active Roster items.
>
> Attributes:
>
> `accountName`
>> if this optional attribute is specified, the operation reads the Roster items from the specified Account.
>
> The Server sends [`rosterItem`](#) data messages for all currently active Roster "contacts".

`rosterSet`
> This operation modifies a Roster "contact".
>
> Attributes:
>
> `accountName`
>> if this optional attribute is specified, the operation updates the Roster in the specified Account.
>
> `peer`
>> the contact E-mail address.
>
> `subscription`
>> an optional attribute:
>>
>> - `update` or no attribute: update the contact information.
>> - `remove`: remove the contact from the Roster.
>> - `subscribed`: confirm the contact request to monitor the user's presence information.
>> - `unsubscribed`: reject the contact request to monitor the user's presence information, or revoke the already granted right to monitor.
>> - `subscribe`: send a request to monitor the contact's presence information.
>> - `subBoth`: confirm the contact request to monitor the user's presence information, and send a request to monitor the contact's presence information.
>> - `unsubscribe`: stop monitoring the contact's presence information.
>
> `name`
>> an optional attribute specifying the Contact real name.
>
> Body:

a set of `group` XML elements; each element body specifies the name of a Group this contact belongs to.

The `update`, `subscribed`, and `subscribe` operations create a new Roster item if the item with the specified E-mail address does not exist.

`rosterGroupSet`
This operation manages Roster "contact groups".

Attributes:
`accountName`
if this optional attribute is specified, the operation updates the Roster in the specified Account.

`group`
the group name.

`mode`

- `add`: create a new contact group with the specified name.
- `delete`: remove the exiting contact group with the specified name from the Roster.
- `rename`: rename the exiting contact group.

`newName`
the new contact group name (used only with the `mode` attribute set to `rename`).

`presenceSet`
This operation sets the user presence information. The Server aggregates the presence information reported by all currently connected clients (XIMSS, XMPP, SIP) and distributes it to all network entities subscribed to that information.

Attributes:
`type`
this optional element may have the `unavailable` value. It indicates that the user is "virtually offline".
If this attribute is absent, the user is online.

Body:
an optional `show` XML element; its text body specifies the XMPP-style user presence status:

- `dnd` - do not disturb, busy, on-the-phone.
- `away` - will be right back, in a meeting, out-to-lunch.
- `xa` - away ("extended away").

Alternatively, you can use a `presence` XML element instead of the `show` XML element. The `presence` XML element text body specifies the more detailed user presence status:

- `offline` - "virtually offline".
- `online` - online.
- `on-phone` - user is engaged into a real-time conversation.
- `in-meeting` - user is in a middle of a meeting.
- `busy` - generic form "cannot chat right now".
- `be-back` - user will be right back
- `out-lunch` - user is out for a break/meal.
- `away` - extended away.

The XML body can also contact an optional `status` XML element; its text body sets the custom status message. To remove a previously set custom status message, specify an empty `status` XML element.

When a session starts, the user is "virtually offline". The client should use this operation to indicate that the user is online.

When the user disconnects, its session status is automatically changed to `unavailable` (offline).

This operation can also be used to send presence information to a specific target (for example, to join an XMPP conference room). To send presence information rather than to set it, use additional attributes:

Attributes:
`peer`
> the E-mail address of the user or a conference room to send the presence information to.

`clientID`
> optional; the name or ID of the peer device to send the presence information to.

The Server sends the following data messages:

`rosterItem`
> These data messages are sent synchronously when the Server processes the [rosterList](#) request. One message is sent for each Roster item ("Contact").
> These data messages can also be sent sent asynchronously (see below).

> Attributes:
> `peer`
> > the contact E-mail address.

> `subscription`

> - `to`: the user can monitor the contact.
> - `from`: the contact can monitor the user.
> - `both`: the user and the contact can monitor each other.
> - `none`: the user and the contact do not monitor each other.

> `ask`
> > this optional attribute has the `subscribe` value if the user has requested a subscription to the contact's presence information, but this request has not been confirmed yet.

> `name`
> > an optional attribute specifying the Contact real name.

> Body:
> > a set of `group` XML elements; each element body specifies the name of a Group this contact belongs to.

> If the [signalBind operation](#) was used to enable Roster and Presence notifications:

> - Every time a Roster item is added or updated, the Server sends `rosterItem` messages with the new or updated data.
> - Every time a Roster item is deleted, the Server sends the `rosterItem` message with the `subscription` attribute set to `remove`.
> - The Server sends the [presence](#) data messages.

`presence`
> These asynchronous data messages are sent when a presence state of some Roster "contact" (item) is changed, and Presence notifications are enabled.

> Attributes:
> `peer`
> > the Contact E-mail address.

> `type`

an optional attribute:

- `unavailable`: the contact is offline.
- `subscribe`: the contact is requesting subscription to the user's presence information.
- *absent*: the contact is online.

Body:
a `presence` XML element; its text body specifies the detailed contact presence status (see above);
an optional `show` XML element; its text body specifies the XMPP-style contact presence status (see above).

These asynchronous data messages are also sent when a remote entity sends its presence information to this particular XIMSS session (for example, a remote conference room informs this session about users joining and leaving the room). In this case, the data message contains additional attributes.

Attributes:
clientID
the name or ID of the peer device the presence information was sent from.

---

# XMPP-style requests

A client can send and receive XMPP-style "iq" requests.

iqSend
This operation sends an XMPP-style "iq" request to the specified user. The Server sends a XIMSS response without waiting for the request to be actually delivered and processed.

Attributes:
peer
the E-mail address of the user to send the iq request to.

clientID
optional; the name or ID of the peer device to send the iq request to.

type
the XMPP iq request type (`get`, `set`, `result`, or `error`).

iqid
this attribute value is used as the `id` attribute of the XMPP iq request.

Body:
the XMPP iq request body.

The Server sends the following data messages:

iqRead
These asynchronous data messages are sent when the Server receives an XMPP-style iq request for the client user.

Attributes:
peer
the sender E-mail address.

clientID

optional; the name or ID of the peer device this message was sent from.

type
>the XMPP iq request type (`get`, `set`, `result`, or `error`).

iqid
>the `id` attribute of the received XMPP iq request.


Body:
>the XMPP iq request body.

---

# Preferences

A client can retrieve and update the authenticated user Preferences and [Public Info](#) data.

prefsRead
>This operation retrieves authenticated user Preferences or Public Info data.

>Attributes:
>type
>>this is an optional parameter.
>>If the specified value is `default`, the default Preferences for the Account Domain are retrieved.
>>If the specified value is `custom`, the explicitly specified Account Preferences are retrieved.
>>If the specified value is `publicInfo`, the Account Public Info settings are retrieved.
>>If the attribute is not specified, the effective Account Preferences are retrieved.

>Body:
>>Zero or more XML `name` elements. These elements body specify the names of elements to retrieve; If no `name` element is specified, all Preference or Public Info elements are retrieved.

>The Server returns one [prefs](#) data message.


prefsStore
>This operation updates authenticated user Preferences or Public Info data.

>Attributes:
>type
>>this is an optional parameter.
>>If the specified value is `publicInfo`, the Account Public Info settings are updated.
>>If the attribute is not specified, the custom Account Preferences are updated.

>Body:
>>XML elements with the names equal to the Preference element names. An element body contains the new Preference element value. If the value is the `default` string, the Public Info or custom Preference value is removed, and the default value becomes the effective element value.
>>If the value is an array, it should be presented using the [array](#) XML presentation.
>>If the value is a dictionary, it should be presented using the [dictionary](#) XML presentation.

>If the body contains the `UserFromAddr` element, the Server uses this element and an optional `UserFromName` element to compose the `UserFrom` element with an E-mail address value (in the "*UserFromName_value*" <*UserFromAddr_value*> format).


prefsReload

The Server implements some operations (such as message removal) based on the Preference values read when the session is started. When some other session modifies the Preferences, the client may use this operation to tell the Server to update these "cached" values.

Attributes:
>    *none*

The Server sends the following data messages:

prefs
>    This synchronous data message is sent when the Server processes the <u>prefsRead</u> request.

>    Attributes:
>    type
>    >    an optional attribute, the same as the request attribute.

>    Body:
>    >    XML elements, with the names equal to Preference or Public Info element names. An element body contains the element value.
>    >    If the value is an array, it is presented using the <u>array</u> XML presentation.
>    >    If the value is a dictionary, it is presented using the <u>dictionary</u> XML presentation.

>    If the prefs data contains the UserFrom string, the Server tries to parse that E-mail address. If the address is parsed, the UserFromAddr element (with the pure userName@domainName address) and, optionally, UserFromName element (with the address comment) are added to the data message body.

prefsModified
>    These asynchronous data messages are sent when the Account Preferences are modified (by this session, some other session, or some other CommuniGate Pro Server component).
>    It it recommended that the client re-reads the Account Preferences in response to this message.

---

# File Storage Operations

A client can manage the Account <u>File Storage</u>.
If the authenticated user has the <u>CanAccessWebSites</u> Domain Administrator right, this client can manage files in someone else's Account by using the prefix when specifying file names: *~accountName/fileName*.

fileList
>    This operation provides a list files in a File Storage directory, and the information about these files.

>    Attributes:
>    directory
>    >    an optional attribute with the File Storage subdirectory name. If absent and the fileName attribute is also absent, the content of the Account top File Storage directory is returned.

>    fileName
>    >    an optional attribute, specified only when the directory attribute is absent. If specified only the information about this File Storage entry is returned, even if it is a directory.

>    The Server returns one <u>fileInfo</u> data message for each file or directory in the specified File Storage directory.

fileDirInfo
>    This operation provides information about all Account files or all files in the specified directory.

Attributes:

directory
an optional attribute with the File Storage subdirectory name. If absent, the information about all Account File Storage files is returned.

The Server returns one `fileDirInfo` data message.

fileDirList
This operation provides a list of all File Storage folders. The Server returns one `fileDirName` data message for each File Storage directory.

fileWrite
This operation stores data in a file.

Attributes:

fileName
the name of file to write to.

position
If this attribute is absent, then this operation completely rewrites the specified file. If the file did not exist, it is created.
If this attribute value is `append`, this operation adds the data to the file end (atomically).
If this attribute value is `new`, this operation first checks that the specified file does not exist, and then creates the file and stores the data in it.
If this attribute value is a number, the file must exist, and the operation rewrites the file from the byte position specified with this number.

type
If this optional attribute exists and its value is `binary`, the request body is `base64` XML element, its body is base64-decoded before being written to the file.
If this optional attribute exists and its value is `object`, the request body is an XML representation of a data Object; the text representation of this Object is written to the file. If this value is used, the `position` attribute must not be specified.
If this optional attribute exists and its value is `xml`, the request body must be a single XML element. Its text representation is written to the file. If this value is used, the `position` attribute must not be specified.
If this optional attribute exists and its value is `vcard`, the request body must be a single XML element representing a vCard object. The vCard text representation is written to the file. If this value is used, the `position` attribute must not be specified.

Body:
Either a text with file data, or a `base64` XML element, or an XML representation of an object, or an XML element to write to the file.

fileStore
This operation stores an uploaded file or a message MIME part as a File Storage file.

Attributes:

fileName
the name of file to be created.

uploadID
a string identifying a file in the "uploaded file set".

folder, UID, partID
these attributes have the same meaning as for the `fileRead` operation. They identify a message MIME part to retrieve, decode, and store as a file.

calendar
use this attribute instead of the `folder` attribute to copy a MIME part from an open Calendar, rather than a E-mail folder.

position
: this optional attribute has the same meaning as for the `fileWrite` operation.

Either a `uploadID` attribute should be specified, or the `folder` is specified, or the `calendar` attribute should be specified. No two of these attributes should be specified together in the same `fileStore` operation.

`fileRead`
: This operation reads data from a file.

Attributes:

`fileName`
: the name of file to read from.

`position`
: This numeric attribute specifies the file position to start reading from. If this attribute is absent, then the file is read from its start (from the file position 0).

`limit`
: This numeric attribute specifies the maximum size of the file data portion to read. If specified, it should not exceed 3MB. If not specified, the 3MB limit is used.

`type`
: If this optional attribute exists and its value is `binary`, the file data is returned base64-encoded, as the `base64` XML element body.
: If this optional attribute exists and its value is `object`, the file data is parsed as a text presentation of an [Object](#), and the Object [XML representation](#) is returned. If this value is used, the `position` and `limit` attributes must not be specified, and the file size should not exceed 3MB.
: If this optional attribute exists and its value is `xml`, the file data is parsed as an XML document, and its XML content is returned. If this value is used, the `position` and `limit` attributes must not be specified, and the file size should not exceed 3MB.
: If this optional attribute exists and its value is `vcard`, the file data is parsed as vCard document, and its XML representation is returned. If this value is used, the `position` and `limit` attributes must not be specified, and the file size should not exceed 3MB.

The Server returns a [fileData](#) data message.

`fileRename`
: This operation renames a file in the File Storage.

Attributes:

`fileName`
: the name of file to rename.

`newName`
: the new name for the file.

`fileRemove`
: This operation removes a file from the File Storage.

Attributes:

`fileName`
: the name of file to remove.

`fileCopy`
: This operation copies file or message contents into a File Storage file.

Attributes:

newName
>     the name of file to create (copy target). If file with this name already exists, it is removed.

fileName
>     optional: the name of File Storage file to copy.

Body:
>     one optional `copyMIME` element. It specified a message part to decode and copy into the specified file.

The operation must contain either a `fileName` attribute or one `copyMIME` body element.

## fileDirCreate
This operation creates a file directory in the File Storage.

Attributes:
fileName
>     the name of file directory to create.

## fileDirRename
This operation renames a file directory in the File Storage.

Attributes:
fileName
>     the name of file directory to rename.

newName
>     the new name for the file directory.

## fileDirRemove
This operation removes a file directory from the File Storage.

Attributes:
fileName
>     the name of file directory to remove. Only empty directories can be removed.

## fileAttrRead
This operation reads file or directory attributes.

Attributes:
fileName
>     the name of file or file directory.

Body:
>     a set of `field` XML elements. Each element should have a text body containing the name of an attribute to retrieve.

If no `field` element is specified, all attributes are retrieved.
The Server returns a `fileAttrData` data message.

## fileAttrWrite
This operation modifies file or directory attributes.

Attributes:
fileName
>     the name of file or file directory.

Body:
>    a set of file attribute XML elements.
>    If a body XML element has the `mode` attribute with the `delete` value, then all attributes with the same name as the name of this body XML element are removed, and the body XML element is not added.
>    If a body XML element has the `mode` attribute with the `add` value, the body XML element is added to the file attributes.
>    If a body XML element has the `mode` attribute with any other value or does not have the `mode` attribute, then all attributes with the same name as the name of this body XML element are removed, and then the body XML element is added to the file attributes.
>    Before a body XML element is added to the attribute set, its `mode` attribute (if any) is removed.

`fileLock`

>    This operation manages file or directory "locks".
>
>    Attributes:
>    `fileName`
>    >    the name of file or file directory.
>
>    `mode`
>    >    the requested operation: `lock` (locking and lock refreshing), `unlock` (unlocking), `testlock` (noop, reading the lock list).
>
>    `scope`
>    >    this attribute is required for the locking operation, it specifies the lock scope: `exclusive` or `shared`.
>
>    `type`
>    >    this attribute is required for the locking operation, it specifies the lock type (`read`, `write`, etc.)
>
>    `token`
>    >    this attribute is required for the lock refreshing and unlocking operations, it specifies the lock token previously received by this client application. The lock belonging to this `owner` with the specified `token` is refreshed or unlocked (removed).
>
>    `timeTill`
>    >    this attribute is required for the locking and lock refreshing operation, it specifies the lock expiration time (specified for the selected time zone). If not specified, the default expiration time is used.
>
>    `create`
>    >    if this attribute is present and set to the `yes` string, the locking operation does not fail if the specified file or directory does not exist, but it tries to create it first.
>
>    Body:
>    >    for the locking operations, the `owner` XML element containing the information about the lock owner.
>
>    If this operation succeeds, the Server sends a <u>fileLockData</u> data message.

`fileSubList`

>    This operation makes the Server send a <u>fileSubscription</u> data message for each element in the Account *file subscription* set. Note that files in this set may or may not exist.

`fileSubUpdate`

>    This operation modifies Account *file subscription* set.
>
>    Body:
>    >    a set of `fileSubscription` elements.
>
>    >    Attributes:
>    >    `fileName`
>    >    >    a file name in the UTF-8 character set.
>
>    >    `mode`

if this attribute value is `add` the `fileName` name is added to the *file subscription* set (unless the set already contains this name).

otherwise, the `fileName` name is removed from the *file subscription* set.

The Server sends the following data messages:

`fileInfo`

This synchronous data message is sent when the Server processes the [fileList](#) request.

Attributes:

`fileName`

the file or subdirectory name.

`directory`

an optional attribute, the same as the `directory` attribute in the [fileList](#) request.

`type`

an optional attribute exists and contains the `directory` value, if this message describes a subdirectory.

`size`

an optional attribute with the file size in bytes (absent if this message describes a subdirectory).

`timeModified`

an optional attribute with the file modification date and time (local time).

`timeAttrModified`

an optional attribute with the file attribute set modification date and time (local time).

If the `fileName` attribute was specified in the [fileList](#) request, or if the name specified in the `directory` request attribute belongs to a regular file, and not to a file directory, the `directory` attribute is not included into the `fileInfo` message, and the `fileName` attribute contains the full name of the target file or directory.

`fileDirInfo`

This synchronous data message is sent when the Server processes the [fileDirInfo](#) request.

**Note:** all attributes are optional, they do not exist, if the system does not maintain the requested values (for example, when the operation is applied to a virtual `$DomainPBXApp$` directory).

Attributes:

`directory`

an optional attribute, the same as the `directory` attribute in the [fileDirInfo](#) request.

`files`

the total number of files in the Account or in the specified subdirectory (and all nested directories).

`size`

the total size of all files (in bytes).

`filesLimit`

the storage file limit setting of the directory owner Account.

`sizeLimit`

the file storage limit setting of the directory owner Account.

`fileDirName`

This synchronous data message is sent when the Server processes the [fileDirList](#) request.

Attributes:

`directory`

the File Storage directory name.

`fileData`

This synchronous data message is sent when the Server processes the [fileRead](#) request.

Attributes:

`fileName`

the name of read file.

`position`

the file position of the read data.

`slice`

the size of read data (in bytes). To read the next portion of the file, add the `position` and `slice` values to get the new `position` value.

`size`

the file total size.

`timeModified`

an optional attribute with the file modification date and time (local time).

`type`

a copy of the request `type` attribute.

Body:

Either a text with file data, or the `base64` (its body is base64-encoded file data), or an XML representation of the file content Object.

Examples:

```
C:<fileRead id="A001" fileName="test.txt" />
S:<fileData id="A001" fileName="test.txt"position="0" slice="22" size="22" timeModified="20061018T115836">This is not
your file!</fileData>
S:<response id="A001"/>

C:<fileRead id="A002" fileName="test.txt" limit="15" />
S:<fileData id="A002" fileName="test.txt"position="0" slice="15" size="22" timeModified="20061018T115836">This is not
you</fileData>
S:<response id="A002"/>

C:<fileRead id="A003" fileName="test.txt" limit="15" position="15" />
S:<fileData id="A003" fileName="test.txt"position="15" slice="7" size="22" timeModified="20061018T115836">r
file!</fileData>
S:<response id="A003"/>

C:<fileRead id="A004" fileName="test.txt" limit="15" position="15" type="binary" />
S:<fileData id="A004" fileName="test.txt"position="15" slice="7" size="22"
timeModified="20061018T115836"><base64>ciBmaWxlIQ==</base64></fileData>
S:<response id="A004"/>

C:<fileRead id="A005" fileName="test.txt" position="25" />
S:<response id="A005" errorText="reading beyond the EOF mark" errorNum="500" />
```

`fileAttrData`

This data message is sent when the Server processes the [fileAttrRead](#) request.

Attributes:

`fileName`

the name of file or file directory.

Body:

a set of file attribute XML elements.

`fileLockData`

These data messages are sent when the Server processes the [fileLock](#) request.

Attributes:

fileName
>    the file name in the UTF-8 character set.

token
>    the token string for the newly created or refreshed lock.

expires
>    the newly created or refreshed lock expiration time.

create
>    this attribute exists if an empty file has been created as a result of this `fileLock` request.

Body:
>    a set of `fileLockData` XML elements for all existing file locks.
>    Each element contains the `expires`, `scope`, `type` attributes described above.
>    The element body contains the `owner` XML element with the lock owner information.

`fileSubscription`
>    These data messages are sent when the Server processes the `fileSubList` request.

>    Attributes:

fileName
>    the file name in the UTF-8 character set.

# Automated Rule Management

A client can manage the Account [Automated Rules](#). Each rule is presented as a `rule` XML element. Rules sets are addressed using the `type` parameter. The following parameter values are supported:

- `mailIn` - incoming [E-mail (Queue) Rules](#).
- `signalIn` - incoming [Signal Rules](#).

Each Rule in a set has name and priority attributes. Signal Rules also have a `stage` attribute, specifying when the Rule should be applied.

Each Rule contains zero or more `condition` elements, zero or more `action` elements, and zero or one `comment` elements:

condition
>    See the [Rules](#) section for more information.

>    Attributes:

opCode
>    the Rule condition data.

operator
>    the Rule condition operation.

>    Body:
>    the Rule condition parameter.

action
>    See the [Rules](#) section for more information.

>    Attributes:

opCode

>   the Rule action operation.


Body:

>   the Rule action parameter.


comment

Body:

>   the Rule comment text.


Example:

a Mail Rule storing all messages with the `X-Junk: 5` header field in the `Junk` Mailbox. Exception is made for messages coming from authenticated users.

```
<rule type="mailIn" name="Junk Filter" priority="2" >
   <condition opCode="Header Field" operator="is">X-Junk: 5*</condition>
   <condition opCode="Source" operator="is not">authenticated</condition>
   <action opCode="Store in">Junk</action>
   <action opCode="Discard"></action>
   <comment>This is my test Rule&#x21;</comment>
</rule>
```

ruleList

> This operation tells the Server to send one [rule](#) data message for each Account Rule of the specified type. These messages do not have a body.
>
> Attributes:
> type
>
> >   the Rule set.


ruleRead

> This operation tells the Server to send one [rule](#) data message for the specified Rule. This message body contains XML elements with the Rule condition(s), action(s), and an optional comment.
>
> Attributes:
> type
>
> >   the Rule set.
>
> name
>
> >   the Rule name.


ruleSet

> This operation stores a new Rule. If there is an existing Rule with the same name, it is removed.
> The user should have a right to specify conditions and actions used in both new and old Rules.
>
> Attributes:
> type
>
> >   the Rule set.
>
> name
>
> >   the Rule name.
>
> priority
>
> >   the Rule priority.
>
> stage
>
> >   the Rule stage (for Signal Rules).

Body:
    the same XML elements as used in the `rule` message: zero or more `condition` elements, zero or more `action` elements, and zero or one `comment` element.

`ruleUpdate`
    This operation modifies the existing Rule priority and/or its stage (for Signal Rules).
    The user should have a right to specify conditions and actions used in the Rule to be modified.

    Attributes:
    `type`
        the Rule set.

    `name`
        the Rule name.

    `priority`
        the Rule priority.

    `stage`
        the Rule stage (for Signal Rules).

`ruleRename`
    This operation renames an existing Rule.
    The user should have a right to specify conditions and actions used in the Rule to be renamed.

    Attributes:
    `type`
        the Rule set.

    `name`
        the existing Rule name.

    `newName`
        the new Rule name.

`ruleRemove`
    This operation removes an existing Rule.
    The user should have a right to specify conditions and actions used in the Rule to be removed.

    Attributes:
    `type`
        the Rule set.

    `name`
        the existing Rule name.

The Server sends the following data messages:

`rule`
    This synchronous data message is sent when the Server processes the `ruleList` request or the `ruleRead` request. See above for the `rule` message format.

# Remote POP, Remote SIP Management

RPOP records are presented using the `rpopRecord` XML elements:

Attributes:
`name`
    the record name

Body:
    the XML represntation of an RPOP Record dictionary.

A client can manage the Account RPOP Account records.

`rpopList`
    This operation tells the Server to send one `rpopRecord` data message for each Account RPOP record.

    Attributes:
    *none*


`rpopUpdate`
    This operation updates the Account RPOP record set.
    The user should have a right to modify the Account RPOP records.

    Attributes:
    *none*

    Body:
        one or more `rpopRecord` elements. These elements must have an additional attribute:

        `mode`
            if this attribute value is `delete`, the only the `name` attribute is used. The specified RPOP record is removed from the Account RPOP record set.
            Otherwise, the RPOP record specified using the `rpopRecord` element body is added to the Account RPOP record set; if a record with the same name existed in the set, it is removed.




RSIP records are presented using the `rsipRecord` XML elements:

Attributes:
`name`
    the record name

Body:
    the XML represntation of an RSIP Record dictionary.

A client can manage the Account RSIP records.

`rsipList`
    This operation tells the Server to send one `rsipRecord` data message for each Account RSIP record.

    Attributes:
    *none*


`rsipUpdate`
    This operation updates the Account RSIP record set.
    The user should have a right to modify the Account RSIP records.

Attributes:
*none*

Body:
> one or more `rsipRecord` elements. These elements must have an additional attribute:

> mode
>> if this attribute value is `delete`, the only the `name` attribute is used. The RSIP record with the same `name` attribute is removed from the Account RSIP record set.
>> Otherwise, the RSIP record specified using the `rsipRecord` element body is added to the Account RSIP record set; if a record with the same name existed in the set, it is removed.

The Server sends the following data messages:

rpopRecord
> This synchronous data message is sent when the Server processes the <u>rpopList</u> request. See above for the <u>rpopRecord</u> message format.
> The optional `timeCompleted` attribute indicates the time (in the selected time zone) when the last connection attempt was made.
> If the last attempt to poll the RPOP account failed, the record has an additional `errorText` attribute containing the failed attempt error code string.

rsipRecord
> This synchronous data message is sent when the Server processes the <u>rsipList</u> request. See above for the <u>rsipRecord</u> message format.
> The optional `timeCompleted` attribute indicates the time (in the selected time zone) when the last connection attempt was made.
> If the last attempt to poll the RSIP account failed, the record has an additional `errorText` attribute containing the failed attempt error code string.

# Real-Time Application Management

A client can interact with the <u>Real-Time Applications</u> running on the CommuniGate Pro Server or Cluster.

A XIMSS session can communicate with <u>Tasks</u> by sending them <u>Events</u>. Tasks see a XIMSS session as a yet another Task, so they can send Events back to it.

Each XIMSS session maintains a list of Tasks Handles for the Tasks it communicates with. Each handle in the list is associated with an taskID string. The XIMSS client uses these taskID strings to address the Tasks in the list.

taskFindMeeting
> This operation implements the FindMeeting operation. See the <u>CG/PL</u> section for more information.

> Attributes:
> meetingSet
>> the optional Meeting set name. If the parameter value is missing or it is an empty string, the Default Meeting Set is used.

> meetingName
>> the unique ID or name for the Meeting.

> If this operation succeeds, the Server returns a <u>taskMeeting</u> data message with the found meeting data.

taskCreateMeeting

This operation implements the CreateMeeting operation. See the [CG/PL](#) section for more information.

Attributes:
meetingSet
    the optional Meeting set name. If the parameter value is missing or it is an empty string, the Default Meeting Set is used.

meetingName
    the unique ID or name for the Meeting.

Body:
    The text to add to the Meeting Event as a parameter.

`taskRemoveMeeting`, `taskClearMeeting`, `taskActivateMeeting`, `taskDeactivateMeeting`
These operations implement the Meeting operations. See the [CG/PL](#) section for more information.

Attributes:
meetingSet
    the optional Meeting set name. If the parameter value is missing or it is an empty string, the Default Meeting Set is used.

meetingName
    the unique ID or name for the Meeting.

`taskSendEvent`
This operation send an Event to a Task.

Attributes:
taskRef
    the internal taskID of the Task to send an Event to.

eventName
    the name of the Event to send.

Body:
    optional text or an XML presentation of a [data object](#) to be sent as the Event parameter.

`taskStart`
This operation starts a [Real-Time Application](#) Task.
If this operation succeeds, the server sends a [taskCreated](#) data message.

Attributes:
programName
    the application name.

entryName
    this optional attribute specified the program entry name; if not specified, the `main` entry is started.

Body:
    a set of `param` XML elements. These element text bodies are placed into the Task `startParameter` array.

`taskClose`
There are some session resources associated with each internal taskID. If your client works with many different Tasks, it is recommended that it uses the taskClose operation when it ends communication with a particular Task, so its session-internal

taskID is released.
If the same Task sends an Event to this session, or its Task Handle is discovered using other mechanisms, the new taskID is assigned to this Task in this session.

Attributes:
`taskRef`
    the internal taskID to release.

The Server sends the following data messages:

`taskMeeting`
    This synchronous data message is sent when the Server processes the `taskFindMeeting` request.

    Attributes:
    `meetingSet, meetingName`
        copies of the `taskFindMeeting` request attributes.

    `taskRef`
        an optional attribute: the internal taskID of the Task that has activated this Meeting.

    `expires`
        an optional attribute with the Meeting expiration date and time (GMT).

    Body:
        The XML presentation of the Meeting parameters.

`taskCreated`
    This synchronous data message is sent when the Server processes the `taskStart` request.

    Attributes:
    `taskRef`
        the internal taskID of the created Task.

`taskEvent`
    This asynchronous data message is sent when some Task or the Server itself sends an Event to the current XIMSS session.

    Attributes:
    `taskRef`
        the internal taskID of the Task that has sent this Event. If this attribute is absent, then the Event was generated and sent by the Server itself.

    `eventName`
        the name of the received Event.

    Body:
        The text or the XML presentation of the Event parameter.

# Directory

A client can access the global Directory.

listDirectory
> This operation retrieves data from the Directory.
>
> Attributes:
> baseDN
>> the directory record DN to start the search from. If the attribute value is $, the baseDN is set to the Directory subtree containing records for the current Account Domain.
>
> filter
>> this optional attribute specifies a search filter in the RFC2254 format.
>
> scope
>> this optional attribute specifies the search type:
>>
>> - base - the baseDN record is retrieved (the filter attribute is ignored)
>> - one (default) - the baseDN immediate subtree is searched (one-level search)
>> - sub - the entire baseDN subtree is searched (multi-level search)
>
> limit
>> this optional attribute specifies the maximum number of directory records to retrieve.
>
> cookie
>> this optional attribute specifies the "paging cookie". It should be an empty string to retrieve the first set of records. It should be set to the value of the cookie attribute of the last returned directoryData data message to retrieve the next set of records.
>
> Body (optional):
>> a set of field XML elements. Each element should have a text body containing the name of an attribute to retrieve. If no field element is specified, all non-service Directory record attributes are retrieved.
>> **Note:** if the mail attribute is explicitly specified, the Server will compose this attribute value for all CommuniGate Pro Object records without this attribute.
>
> The Server sends a directoryData data message for each record found.

The Server sends the following data messages:

directoryData
> This synchronous data message is sent for each Directory record retrieved with the listDirectory command.
>
> Attributes:
> name
>> the record DN.
>
> cookie
>> exists only in the last returned record, only if the command contained this attribute, and if the retrieved records are not the last records. Use the value of this attribute in the repeated listDirectory command to retrieve more records.
>
> Body:
>> The XML presentation of the Record attributes.

# Datasets

A client can manage the Account Datasets (such as the RepliedAddresses Dataset).

To access Datasets in other Accounts, the dataset name should be specified as *~accountName/datasetName* or *~accountName@domainName/datasetName*

`datasetCreate`

This operation creates an Account dataset.

Attributes:
`dataset`

the name of the dataset to create.

`datasetRemove`

This operation removes an Account dataset.

Attributes:
`dataset`

the name of the dataset to remove.

`ifExists`

if this optional attribute exists, and its value is `yes`, no error is generated when the specified dataset does not exist.

`datasetAddAddress`

This operation adds an address to an Account dataset.

Attributes:
`dataset`

the name of the dataset to add an E-mail address to.

`realName`

this optional attribute specified the E-mail "real name".

Body:

The E-mail address string.

`datasetList`

This operation retrieves entries from an Account dataset.

Attributes:
`dataset`

the dataset name.

`filterField`, `filter`

these optional attributes specify an entry attribute name and value. If specified, the operation returns only the entries that have the specified attribute with the specified value.

`mode`

this optional attribute can be specified only if the `filterField` attribute is specified.
If the `mode` attribute is absent, or if its value is `eq`, then an entry is listed if its entry attribute value is equal to the specified value.
If the `mode` attribute value is `beg`, then an entry is listed if its entry attribute value starts with the specified value.
If the `mode` attribute value is `end`, then an entry is listed if its entry attribute value ends with the specified value.
If the `mode` attribute value is `contains`, then an entry is listed if its entry attribute value contains the specified value as its substring.

The Server sends a <u>datasetData</u> data message for each record found.

`datasetListSubsets`

This operation lists all subsets of the specified Account dataset.

Attributes:
`dataset`
>
> the dataset name. specify an empty string to list the top-level Account datasets.

The Server sends a `datasetSubset` data message for each dataset found.

`datasetSet`

This operation modifies entries in an Account dataset.

Attributes:
`dataset`
>
> the name of the dataset to modify.

Body:
> The `datasetData` element. Its `name` attribute specifies the name of the dataset entry to modify or create. If not specified, a random name is selected for a new entry.
> The element body should be an XML presentation of the entry attributes to modify.

`datasetDelete`

This operation removes an entry from an Account dataset.

Attributes:
`dataset`
>
> the name of the dataset to modify.

`name`
>
> the name of the dataset entry to remove.

The Server sends the following data messages:

`datasetData`

This synchronous data message is sent for each dataset entry retrieved with the `datasetList` request.

Attributes:
`name`
>
> the dataset entry name.

Body:
> The XML presentation of the dataset entry attributes.

`datasetSubset`

This synchronous data message is sent for each dataset subset retrieved with the `datasetListSubsets` request.

Attributes:
`dataset`
>
> the subset entry name.

Body:
> none

# Billing

A client can manage the [Account Balances](#).

`balance`
> This operation performs a [Billing](#) operation.
>
> Attributes:
> *accountName*
>> the target account name (optional). If not specified, the operation is applied to the session owner Account.
>
> *localTime*
>> if this attribute does not exist or its value is not `no`, then all timestamp-type values in the operation parameters should be specified as local times in the Time Zone selected for the current user, and all timestamp-type values in the operation result are converted into the local time in that Time Zone.
>> If this attribute value is `no`, then all timestamp-type values in the operation parameters and result are GMT times.
>
> Body:
>> The XML presentation of a dictionary. The dictionary contains the Billing operation (the `op` element) and the operation parameters.
>
> If the operation produces any result, the Server sends a [balanceData](#) data message.

The Server sends the following data messages:

`balanceData`
> This synchronous data message is sent when a [balance](#) operation produces a result.
>
> Attributes:
> *accountName*
>> the target account name (optional, the same as specified with the `balance` operation request).
>
> Body:
>> The [XML presentation](#) presentation of the result dictionary.

---

# Application Helpers

A client can call [External Application Helpers](#).

`callHelper`
> This operation calls an External Application Helper. If the operation produces any result, the Server sends a [helperReply](#) data message.
>
> Attributes:
> *name*
>> the name of the Application Helper to call.
>
> Body:
>> (optional) the XML presentation of the request sent to the Helper.

The Server sends the following data messages:

`helperReply`
> This synchronous data message is sent when a <u>callHelper</u> operation produces a result.
>
> Attributes:
> *name*
>> the same value as in the `callHelper` operation.
>
> Body:
>> (optional) the XML presentation of the response data received from the Helper.

---

# Banner Retrieval

A client can employ an <u>External Banner System</u>.

`bannerRead`
> This operation retrieves banner info. If the operation produces any result, the Server sends a <u>banner</u> data message.
>
> Attributes:
> *type*
>> the banner type to retrieve (application-specific, such as `prontoEmailTop`, `myClientLeftBanner`).
>
> Body:
>> (optional) the XML presentation of the External Banner System parameter object.

The Server sends the following data messages:

`banner`
> This synchronous data message is sent when a <u>bannerRead</u> operation produces a result.
>
> Attributes:
> *type*
>> the same value as in the `bannerRead` operation.
>
> Body:
>> (optional) the XML presentation of the External Banner System resulting object.

---

# XML Data Formats

Data elements are presented in the XML format using the following conventions.

**EMail**

This XML element represents an E-mail message or its message/rfc822 MIME subpart.

Body:

a set of XML elements containing E-mail message header fields, such as `From`, `Date`, etc., and not including MIME content-related fields (such as `Content-Type`),

and (optionally) a `MIME` element with the message body.

The type of each XML element is the field name:

```
<Subject>Hello, world!</Subject>
```

Field categories:

*Addresses*: `From`, `To`, `Cc`, `Bcc`, `Return-Path`, `Sender`, `Reply-To`, `Disposition-Notification-To`, `Recent-From`, `Recent-To`, `Return-Receipt-To`, `Errors-To`

The element body is the E-mail address, in the *userName@domainName* format, or a more generic *userName*[*%domainA*[*%domainB*]]*@domainName* format.

These elements can contain the `realName` attribute - a MIME-decoded address *name-part* or *comment*.

If a field contains several addresses, then several XML elements are created, so each element contains exactly one address.

*Timestamps*: `Date`, `Resent-Date`

These elements contain:

the `timeShift` attribute - the time difference (in seconds) between the local time specified in the field and the corresponding GMT time

the `localTime` attribute - the field time value (in the iCalendar format) in the Time Zone selected for the current user.

The element body is the field global (GMT) time value in the iCalendar format.

`Pty`

These elements body are `High` or `Low` strings. These values are converted to and from numeric `X-Priority` header field values.

*Phones*: `X-Telnum`

The element body is a phone number.

These elements can contain the `type` attribute (`WORK`, `CELL`, `HOME`, `AGENT`, etc.) specifying the type of the phone number/address.

If a field contains several phone numbers, then several XML elements are created, so each element contains exactly one phone number.

*Unstructured*

All fields not listed in previous categories belong to this category.

The element body contains the MIME-decoded field value.

Example:

```
From: "Mr. Sender." <user1@example.com>
To: user2@example.com (My Friend),
 =?iso-8859-1?Q?=4Eot=20A=20Friend?= <user2@example.com>
Date: Mon, 10 Apr 2006 13:15:48 -0700
Subject: It's 1:15PM now, the meeting has started!
```

```
<EMail>
   <From realName="Mr. Sender.">user1@example.com</From>
   <To realName="My Friend">user2@example.com</To>
   <To realName="Not A Friend">user3@example.com</To>
   <Bcc>user1@example.com</Bcc>
   <Date timeShift="-25200" localTime="20060410T151548">20060410T201548Z</Date>
   <Subject>It's 1:15PM now, the meeting has started!</Subject>
</EMail>
```

## MIME

This XML element represents a message body or its part (referred to as "part" in the rest of this section).

Attributes:

`partID`

this string provides the MIME part location within the message. It can be used to retrieve this message part.

`estimatedSize`

the estimated size (in bytes) of the part data, after the part data is decoded.

`type`

the part `Content-Type` type, without the subtype information.

subtype
>    the part `Content-Type` subtype, if present.
charset
>    the part character set (assume UTF-8 if the attribute is absent).
contentID
>    the `Content-ID` string (without the enclosing angle brackets).
disposition
>    the `Content-Disposition` string (without parameters).
description
>    the `Content-Description` string.
location
>    the `Content-Location` string.
class
>    the `Content-Class` string, after removing the `urn:` and `content-classes:` prefixes.
Type-*name*
>    any Content-Type field *name* parameter with its value, excluding the `boundary`, `charset`, and `format` parameters.
Disposition-*name*
>    any Content-Disposition field *name* parameter with its value.

Body:
>    The element body is optional. When present, it contains the part body data.
>
>    If the Server fails to read the element body part data, the `readError` attribute is added to the element. The attribute value contains the reading error code.
>
>    If the Server fails to parse the element body part data, the `parserError` attribute is added to the element. The attribute value contains the parsing error code.
>
>    The element format depends on the part `Content-Type` (the "*" symbol below means any Content-Type or subtype):

multipart/*
>    Zero or more `MIME` XML elements containing message subparts.
message/rfc822
>    A `EMail` XML element for the enclosed message.
text/rfc822-headers
>    An `EMail` XML element (not containing any `MIME` body element).
text/directory, text/x-vcard
>    Zero or more `vCard` XML elements.
text/x-vgroup
>    One `vCardGroup` XML element.
text/calendar
>    an `iCalendar` XML element.
message/disposition-notification, message/delivery-status
>    a `MIMEReport` XML element.
*/xml, */*+xml
>    The XML data of the body element.
text/*
>    The decoded message text, with all EOL symbols replaced with the Line Feed (0x0A) symbol.

**Note:** When a message is being retrieved with the `folderRead` operation, a MIME element body is included only if:

- its Content-Type is one of the supported types listed above
- its Content-Disposition is not `attachment`
- its size plus the size of the parts already included into the response XML does not exceed the `totalSizeLimit` operation attribute.

Example:

```
 From: <user1@example.com>
To: user2@example.com
MIME-Version: 1.0
Content-Type: multipart/alternative;boundary="abcd"
Content-Description: Test Message

--abcd
Content-Type: text/plain; charset="iso-8859-1";
format=flowed; paramX="valueX"
Content-Transfer-Encoding: quoted-printable

=46rom where I stay, I can see & hear a lot!

--abcd
Content-Type: text/html; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable
```

```
<html><body bgcolor=3D"yellow">
<I>From where I stay</I>, I can see &amp; hear a lot!
</body></html>
--abcd--
```

```
<EMail>
   <From>user1@example.com</From>
   <To>user2@example.com</To>
   <MIME Type="multipart" subtype="alternative" Description="Test Message">
     <MIME type="text" subtype="plain" charset="iso-8859-1" type-paramX="valueX">
       From where I stay, I can see &amp; hear a lot!
     </MIME>
     <MIME type="text" subtype="html" charset="iso-8859-1" type-paramX="valueX">
     &lt;html&gt;&lt;body bgcolor=&quot;yellow&quot;&gt;
     &lt;I&gt;From where I stay&lt;/I&gt;, I can see &amp;amp; hear a lot!
     &lt;/body&gt;&lt;/html&gt;
     </MIME>
   </MIME>
</EMail>
```

## MIMEReport

This XML element represents a message report, such as Disposition Notification or a Delivery report.

Attributes:
> *none*

Body:
> a set of XML elements containing report header fields, such as Reporting-MTA, Final-Recipient, etc.
> and (optionally) MIMEReport elements for the report body.


Example:
```
Subject: Delivery report: TEST - disposition and delivery
From: <MAILER-DAEMON@remote.example.com>
To: <sender@local.example.com>
Date: Wed, 02 May 2007 00:33:13 -0700
Message-ID: <receipt-39457791@remote.example.com>
X-MAPI-Message-Class: REPORT.IPM.Note.DR
MIME-Version: 1.0
Content-Type: multipart/report; report-type="delivery-status"; boundary="_===39457791====remote.example.com===_"


--_===39457791====remote.example.com===_
Content-Type: text/plain; charset="utf-8"

Message delivered to '<recepient@remote.example.com>'
LOCAL module(account recepient) reports:
Delivered to the user mailbox


--_===39457791====remote.example.com===_
Content-Type: message/delivery-status

Reporting-MTA: dns; remote.example.com

Original-Recipient: rfc822;<recepient@remote.example.com>
Final-Recipient: LOCAL;<recepient>
Action: delivered
Status: 2.0.0

--_===39457791====remote.example.com===_
Content-Type: text/rfc822-headers

From: "Sender Name" <sender@local.example.com>
Subject: TEST - disposition and delivery
To: recepient@remote.example.com
X-Mailer: CommuniGate Pro WebUser v5.1.9
Date: Wed, 02 May 2007 00:35:51 -0700
Message-ID: <web-3990004@local.example.com>
MIME-Version: 1.0
Disposition-Notification-To: <sender@local.example.com>
Content-Type: text/plain;charset="utf-8";format="flowed"
Content-Transfer-Encoding: 8bit

--_===39457791====remote.example.com===_--
```

```
<EMail>
  <Subject>Delivery report: TEST - disposition and delivery</Subject>
  <From>MAILER-DAEMON@stalker.com</From>
  <To>sender@local.example.com</To>
  <Date localTime="20070502T003313" timeShift="-25200">20070502T073313Z</Date>
  <Message-ID>&lt;receipt-39457791@remote.example.com&gt;</Message-ID>
  <X-MAPI-Message-Class>REPORT.IPM.Note.DR</X-MAPI-Message-Class>
  <MIME estimatedSize="1433" subtype="report" type="multipart" Type-report-type="delivery-status">
   <MIME charset="utf-8" estimatedSize="120" partID="01" subtype="plain" type="text">Message delivered to
&#39;&lt;recepient@remote.example.com&gt;&#39;
LOCAL module(account recepient) reports:
 Delivered to the user mailbox
```

```
    </MIME>
    <MIME estimatedSize="158" partID="02" subtype="delivery-status" type="message">
     <MIMEReport>
      <Reporting-MTA>dns; remote.example.com</Reporting-MTA>
      <MIMEReport>
       <Original-Recipient>rfc822;&lt;recepient@remote.example.com&gt;</Original-Recipient>
       <Final-Recipient>LOCAL;&lt;recepient&gt;</Final-Recipient>
       <Action>delivered</Action>
       <Status>2.0.0</Status>
      </MIMEReport>
     </MIMEReport>
    </MIME>
    <MIME estimatedSize="787" partID="03" subtype="rfc822-headers" type="text">
     <EMail>
      <From realName="Sender Name">sender@local.example.com</From>
      <Subject>TEST - disposition and delivery</Subject>
      <To>recepient@remote.example.com</To>
      <X-Mailer>CommuniGate Pro WebUser v5.1.9</X-Mailer>
      <Date localTime="20070502T003551" timeShift="-25200">20070502T073551Z</Date>
      <Message-ID>&lt;web-3990004@local.example.com&gt;</Message-ID>
      <Disposition-Notification-To>sender@local.example.com</Disposition-Notification-To>
     </EMail>
    </MIME>
   </MIME>
  </MIME>
</EMail>
```

# HTTP Access

Some clients may be unable to implement all operations via the XIMSS protocol. When a XIMSS session is created, its ID is reported back to the client using the <u>session</u> data message. The Server provides access to that session via the HTTP protocol.

## Message Part Retrieval

The following URL can be used to retrieve any part of a message visible in any currently opened Folder:

> /Session/*sessionID*/MIME/*folder*/*UID-partID-suffix*[/*filename*]

where

*sessionID*
> the current XIMSS session ID, sent with the <u>session</u> data message

*folder*
> the target Folder name or the target Calendar name

*UID*
> the target message UID in the target Folder or Calendar

*partID*
> the id of the requested message MIME part. This is the string reported in the MIME XML element when the message was retrieved using the folderRead operation. If the entire message is to be retrieved, the *partID* string and the following - symbol should be omitted.

*suffix*
> the retrieval mode:

- P.txt - the entire undecoded part (including the headers and the body) is retrieved.
- H.txt - the part headers are retrieved.
- B.*extension* - the decoded part body is retrieved. You can use any appropriate file name extension. The HTTP response gets the Content-Type of the retrieved MIME part.
- R/*cid* - the *cid* string should be URI-encoded. Its decoded value specifies a MIME-part Content-ID. The Server searches all MIME parts "related" to the current one, and retrieves the body of the part with the matching Content-ID.

  This feature is used to convert cid:-type URLs in HTML-formatted messages.

*filename*
> an optional file name. It is ignored by the Server, but it can help the user browser to store the file under the proper name.

Examples:

```
C:GET /Session/1-2xklkdld8-djdkjk/MIME/INBOX/567-01-B.gif HTTP/1.1
```

```
C:GET /Session/1-2xklkdld8-djdkjk/MIME/INBOX/567-02-01-B.gif/Logo.gif HTTP/1.1
```

It is possible to retrieve only a portion of a message part, using the HTTP `Range` header field. Alternatively (when HTTP request fields cannot be specified), the URL parameters can be used:

`OffsetFrom`
  part data byte-position to start with (0 means to start from the beginning).
`OffsetTill`
  (optional) part data byte-position following the last byte to send.

Example:
```
C:GET /Session/1-2xklkdld8-djdkjk/MIME/INBOX/567-01-B.gif?OffsetFrom=100&OffsetTill=200 HTTP/1.1
```
This request retrieves 100 bytes (bytes 100..199 inclusive) from the GIF file encoded in the specified MIME part.

## Profile Retrieval

The following URL can be used to retrieve the Profile vCard or any of its properties:

> `/Session/`*sessionID*`/PROFILE/[`*propertyname*`[/`*index*`][/`*filename*`]]`

where

*sessionID*
  the current XIMSS session ID, sent with the <u>session</u> data message
*propertyname*
  the vCard property name, such as `PHOTO`. If the property name is not specified, the entire vCard is retrieved.
*index*
  if the vCard contains several values for the specified property name, the index of the value to retrieve (zero-based). If not specified, the index value of 0 is assumed.
*filename*
  any properly encoded URL-string.

Example:
```
C:GET /Session/1-2xklkdld8-djdkjk/PROFILE/PHOTO/0/avatar HTTP/1.1
```
This request retrieves the first value of the "PHOTO" property of the Profile vCard.

## File Uploading

The following URL can be used to upload a file to the session "uploaded file set".

> `/Session/`*sessionID*`/UPLOAD/`*uploadID*

where

*sessionID*
  the current XIMSS session ID, sent with the <u>session</u> data message.
*uploadID*
  a string identifying this file in the "uploaded file set".

The HTTP request should be:

- a POST request, with the multipart/form-data content, and the file (raw binary data) should be sent as the `fileData` form element, or
- a PUT request, and the file (raw binary data) should be sent as the request body.

When the file is uploaded and added to the "uploaded file set", the Server returns the 200 response code. If uploading failed, the Server returns the 500 response code.

The uploaded file is stored together with its Content-Type value.

If the "uploaded file set" already contains a file with the specified `uploadID` value, the old file is removed.

# File Downloading

The following URL can be used to download a file from the session Account File Storage.

        /Session/*sessionID*/DOWNLOAD/*fileName*

where

*sessionID*
        the current XIMSS session ID, sent with the `session` data message.
*fileName*
        the file name in the session Account File Storage.
        Note: to download a file from the "uploaded file set", specify the *fileName* as `$UPLOAD$/`*uploadId*, where *uploadId* is the "uploaded file set" file identifier.

# Data Export

The following URL can be used to export data from a folder or a calendar:

        /Session/*sessionID*/Export/*folderName*/*fileName*[?oldStyle=1]

If the *folderName* is a name of an open Calendar object, the request retrieves an iCalendar text that includes all object VEVENT and VTODO items, and all VTIMEZONE objects referenced.
If the *folderName* is a name of an open folder object, the request retrieves a vCard text that includes all VCard items stored in that folder. If the oldStyle parameter is specified, the vCard items are exported using the vCard 2.1 format and the UTF-8 character set, otherwise the vCard 3.0 format and the Unicode character set is used.

# Private Key and Certificate Export

The following URL can be used to export the Account Private Key and Certificate as a PKCS12 ("PFX") file:

        /Session/*sessionID*/CertAndKey.pfx?PFXPassword=*password*[&PFXPassword1=*password1*]

If the `PFXPassword1` is specified, its value must be the same as the `PFXPassword` value.
The session must have its S/MIME unlocked.
The Account Private Key and Certificate are placed into PKCS12-formated data encrypted using the `PFXPassword` value.

# Abnormal Termination

The following URL can be used to close the session without doing any "logout cleanup" procedures:

        /Session/*sessionID*/KILL

# Web Applications

The CommuniGate Pro Web Application module provides access to various CommuniGate Pro Objects (accounts, messages, mailing lists, web files) via any Web (HTTP/HTML/WML/cHTML) browser.

The HTTP module receives HTTP client browser requests that come to the WebUser port(s), and passes those requests to the Web Application module. The Web Application module either retrieves the requested file, or it starts some internal *web application* code and converts the result into the HTML, WML or other *markup* format. The result is returned to the HTTP module that delivers it back to the client browser.

Read this section if you want to customize your CommuniGate Pro Server WebUser Interface.

# Stateless and Session-based Processing

Regular HTTP servers are *stateless* processors: a user's browser may send several sequential requests, but the HTTP server does not keep any information about the browser or the client between the requests. Each request is processed individually.

The Web Application Server allows users to "log in", providing a name of some CommuniGate Pro Account and the account password. For each successful login, a *Session* is started. The session keeps the information about user actions and requests, so all HTTP requests sent to the same session can share and use the same set of session data.

To maintain a session, all session requests have URLs in the following form:

```
http://hostname:serverport/Session/sessionID/sessionRequest
```

where the *sessionID* string identifies the session, and the *sessionRequest* is the name of the file to retrieve or the application

component to run.

The Web Application Sessions have time-out counters. If no HTTP request has been sent to a session during the configurable time-out interval, the Session is closed. The session user can close the session by sending a request for the special *Bye* page.

# Skins

The Server WebUser Interface implements Skins. Each Skin is a set of files that define how the information is presented to users. Skins files include:

- [WSSP files](#) used to build "web pages" (HTML, WML, etc.) and format the page data.
- static service files - graphic elements, style sheets, etc.
- [Language files](#) containing various static text strings (messages, page and button titles, tags, etc.) referenced from WSSP files.

The CommuniGate Pro software comes with one *Unnamed* Stock Skin, providing the very basic and simple HTML and WML interface. It also comes with some *Named* Stock Skins that provide more visually rich interfaces. This Stock Skins are stored in the *application directory*, they are parts of the software package, and they **should not be modified by server administrators**.

CommuniGate Pro installations can also use Unnamed and Named *custom* Skins. Custom Skins can be created as Server-wide Skins. These custom Skins are available to all users. Each CommuniGate Pro Domain can also have its own set of custom Skins, available only to the users of that Domain.

When a user connects to the WebUser Interface service (the HTTP User port), the *hostname* string specified in a stateless request URL is used to find the CommuniGate Pro [Domain](#). When the Domain is found, its Default Account WebUser Preferences are retrieved and the SkinName (Layout) and Language Settings are used.

The SkinName Setting specified the name of the Skin to use (if that Setting is empty, the Unnamed Skin is used).

If the Skin with the specified Name is not found in the set of the Domain Skins, the Server-wide Skin sets are checked. If the Skin with the specified name is still not found, the Stock Skin with this name is used. If the Named Stock Skin is not found either, the Unnamed Stock Skin is used.

Since Domains can have their own Skin sets, the same request sent to different Domains can display different pages: the Default WebUser Preferences can specified different Skin Names in different Domains, and even if the Preferences are the same, different Skins with the same name can exist in different Domains.

Stateless requests can use any Skin available for the addressed Domain. To use an alternative Skin, a Stateless HTTP request should specify the Skin name using the `Skin` request parameter.

The Language Setting retrieved from the Domain effective Default Account WebUser Preferences specifies the language to use on the stateless page. To use an alternative language, a Stateless HTTP request should specify the language name using the `Language` request parameter.

Session-based HTTP requests do not use the hostname string specified in the URL. Instead, when a user logs in, and a Web Application session is created for the specified CommuniGate Pro [Account](#), the effective Account WebUser preferences are retrieved. Those preferences contain the name of the Skin to use (an empty value specifies an Unnamed Skin). The Skin with the specified name is selected from the Skin set of the Account Domain (note that the Domain specified with the request URL can be different).
If the Account Domain does not have the specified Skin, the Server-wide Skin is used.

Session-based HTTP requests use the Language specified with the Account WebUser Preferences.

## WAP/WML Skins

The HTTP module checks the content of the `Accept` request header field. If this field contains a `wml` substring, the module assumes that the request comes from a WML browser.

Requests coming from WML browsers are processed using WML Skins. For Stateless requests the name of the Skin to use is taken from the WAP/WML Layout parameter of the Default WebUser Preferences for the addressed Domain. When a user logs in using a WML browser, the name of the Session Skin to use is taken from the WAP/WML Layout parameter of the WebUser Preferences for the user Account.

All Skins with names starting with letters `WML` are considered to be WML Skins. These Skins appear in the list of available Skins for the WAP/WML Layout parameters, and these Skins are removed from the list of available Skins for the regular Layout parameters.

## cHTML (I-Mode) Skins

The HTTP module checks the content of the `User-Agent` request header field.
If this field contains a `DoCoMo` substring, the module assumes that the request comes from a Japanese I-Mode browser.
If this field contains a `portalmmm` substring, the module assumes that the request comes from a European I-Mode browser.

I-Mode requests are processed with special I-Mode Skins in the same way as WML requests are processed with WML Skins. The special I-Mode/cHTML WebUser Preferences parameter is used to specify the name of the I-Mode Skin to use.

All Skins with names starting with letters `IMode` are considered to be cHTML Skins. These Skins appear in the list of available Skins for the I-Mode/cHTML Layout parameters, and these Skins are removed from the list of available Skins for the regular Layout parameters.

For all pages retrieved for Japanese cHTML browsers the charset is set to `Shift-JIS`, and all pages are converted into that charset.
For all pages retrieved for European cHTML browsers the charset is set to `windows-1252`, and all pages are converted into that charset.

# Skin Files Hierarchy

When a WebUser Interface request is processed, the Server needs to retrieve certain files from the selected Skin. If the requested file is not found in the selected Skin, and the selected Skin is a Domain Skin, the file is retrieved from the Server-wide Skin with the same name. If the requested file is not found in the Server-wide Skin, the Stock Skin with the same name is checked. If the file is still not found, and the selected Skin is a Named Skin, then unnamed Server-wide and unnamed Stock Skins are checked.

Initially, when no Domain has any custom Skin, and the Server-wide Skins are empty, all Domains can use the Stock Skins only. The Unnamed Skin is selected by default.
By uploading files into the Server-wide Unnamed Skin, the Server Administrator can "shadow" the Unnamed Stock Skin files, and this can change the application look and feel for all Domains.
By uploading files into the Unnamed Skin of some CommuniGate Pro Domain, the Server or Domain Administrator can "shadow" the Stock Skin and Server-wide Unnamed Skin files and change the look and feel for that particular Domain.

This hierarchy allows Server and Domain Administrators to use new Skins that are designed "from scratch", or to develop small Skin variation, re-using the already existing Skins.

The Dynamic Cluster installations have two sets of the Server-wide Skins - one set is used for local Server Domains, while the other, Cluster set is used for all Shared Domains in the Cluster. Modifications of these Cluster-wide Skins, as well as modifications of any Skin in any Shared Domain are automatically distributed to all Cluster Members.

# Languages and Skin Text Dataset

Each Skin can have a Text Dataset - the `strings.data` text file. This "default language" file contains a dictionary. The WSSP script pages can use various commands to retrieve data from this Text Dataset dictionary.

A Skin can contain additional, localized Text Dataset files - *language*`.data` files, where *language* is the language name (*french.data*, *japanese.data*, etc.). If a non-default language is selected, the Text Dataset from the specified language file is used.

When the selected Text Dataset of the selected Skin does not contain the requested data, the same language Text Datasets are checked in all Skin "parents" (as specified above). If the requested data is still not found, the "default language" Text Dataset is used.

Use the UTF-8 character set to place non-ASCII symbols strings into a Text Dataset file. Check the http://www.unicode.org site to learn more about Unicode and the UTF-8 charset.

You may want to break one Text Dataset file into several files. You can place several `strings.`*`subname`*`.data` files into a Skin. These files are read when the `strings.data` file is read. The file content (a dictionary) is added to the Text Dataset dictionary as the *`subname`* key value.
The *`language`*`.`*`subname`*`.data` files are processed in the same way, extending the localized Text Datasets.

---

# Serving Regular Files

When a URL specifies a file with any file name extension other than `.wssp`, the Web Application module retrieves this file from the selected Skin, places it into the internal Skin Cache, and returns that file to the client browser via the HTTP module connection.

The specified file names are always converted into the lowercase letters.

When the Web Application module receives a request for the same file, it is retrieved from the Skin Cache.

If the file has been requested using a Session-based URL, the session time-out counter is reset. This can be used to create a frame in the client browser window, and make that frame periodically retrieve some file using the session URL. As a result, this session inactivity timer can be reset to keep the session alive as long as this frame is displayed in the user's browser.

System and Domain administrators can upload custom files into server-wide and Domain Skins to modify the Web Application look and feel.
For example, the Stock Skin uses the `Logo.gif` file for most of its pages. By uploading a custom Logo.gif file to a server-wide Unnamed Skin you can change the look of the Web Application pages even without creating and uploading custom page (WSSP) files.

To include a file reference to into a .wssp page retrieved with a Stateless request, use the %%filesRef%% prefix in the .wssp code:

```
... href="%%filesRef%%filename.extension" ...
```
See the Code Components for Stateless Requests section for more details.

Sessions can use Named Skins, and the session-based pages usually need to refer to regular files in the same Skin. References in the "session realm" (`href="`*`filename.extension`*`"` or `href="/Session/`*`sessionID`*`/`*`filename.extension`*`"`) work, but they do not allow client browsers to cache these files between sessions, since each session has its own sessionID, and file URLs are different for each session. To allow client browsers to cache regular files, use the %%SESSION(filesRef)%% prefix for file URLs:

```
... href="%%SESSION(filesRef)%%/filename.extension" ...
```
See the Session Dataset description for more details.

---

# Serving Web Application (WSSP) Files

When a URL specifies a resource with the `.wssp` file name extension, the Web Application module retrieves the specified WSSP file from the Skin, and Compiles it into some internal code. The module then runs the Web Application code associated with the file name. This code produces a dataset with various string, array, and dictionary data. Then the module runs the WSSP internal (compiled) code to produce an HTML page using this dataset, and returns the resulting HTML page to the browser using the HTTP module connection.

The specified resource names are always converted into the lowercase letters.

The WSSP Scripting section explains the WSSP file format. System and Domain administrators can create custom WSSP files and upload them to the server-wide and Domain Skins to modify the Web Application look and feel.

The section below lists the available Web Application code components, defining the set of WSSP pages that this version of CommuniGate Pro server can generate. It specifies how each component processes the form parameters sent to it, and what data is included into the dataset it generates.

# Creating and Managing Skins

You can create and manage Skins using the WebAdmin Interface, CLI/API, or any client/protocol (such as FTP) that can access the Account File Storage.

## WebAdmin Editor

The WebAdmin Interface provides Skin Editor pages to manage Server-wide, Cluster-wide, and Domain Skins.

To manage the Server-wide and Cluster-wide Skins, open the Users realm of the WebAdmin Interface, and click the Skins link.

To manage the Domain Skins, open that Domain page in the Users realm of the WebAdmin Interface, and click the Skins link. The Domain Administrator should have the CanModifySkins Access Right to be able to create and modify the Domain Skins.

When the Domain Skins Editor page is opened, and there is no Unnamed Skin for the Domain, the page contains the Create Unnamed Skin button. Click this button to create the Unnamed Skin for this Domain.

The Skin Editor page contains the list of all files "visible" in this Skin: it lists files directly uploaded to this particular Skin, as well as all files uploaded to the Skins used as the "default files" source for this Skin:

| | no file selected | | |
|---|---|---|---|
| | **Name** | **Size** | **Modified** |
| | Help.gif | 155 | 25-Sep-06 |
| default | addressbook.wssi | 1143 | 23-Sep-06 |
| default | alerts.wssp | 1727 | 26-Sep-07 |
| default | answeredletter.gif | 890 | 27-Feb-06 |
| default | attachedFile.gif | 1147 | 27-Feb-07 |
| | ... | | |
| default | mailbox.wssp | 5806 | 28-Sep-06 |
| | mailboxes.wssp | 3452 | 02-Oct-06 |
| | ... | | |
| default | strings.data | 28K | 27-Oct-06 |
| default | german.data | 31K | 28-Oct-06 |
| | ... | | |
| default | website.wssp | 592 | 28-Sep-06 |
| default | websitebody.wssi | 2648 | 28-Sep-06 |

Files directly uploaded to the Skin have a checkbox in the Marker column. Files from the other skins "visible" in this Skin have the word default in that column.

You can download any of the Skin files by clicking the file name.

You can upload a file to the Skin by clicking the Browse button and selecting a file on your workstation, then clicking the Upload button.

You can delete any of the files uploaded to the Skin by selecting the checkboxes and clicking the Delete Marked button.

If you are uploading a .wssp or a .wssi file, the Editor tries to compile that WSSP file first.
If you are uploading a .wcgp or a .wcgi file, the Editor tries to compile that CG/PL file first.

If you are uploading a `.data` file, the Editor tries to parse it as a [dictionary](#) file first.
If the compiler or the parser detects an error, the file is not uploaded, and the file content is displayed on the Editor page, with the red `<--ERROR-->` marker indicating the location of the error.

When you upload a file to any Skin, that Skin cache is automatically cleared. If you upload a file to a Shared Domain Skin or to a Cluster-wide Skin, the update automatically propagates to all Cluster Members.

You can upload a set of files by uploading a TAR-archive (a file with `.tar` name extension). For example, when you have a TAR-archive with a predesigned Skin you can open the Skin you want to modify (the Server-wide Unnamed Skin or Domain-wide Unnamed Skin or some Named Skin), and upload the `.tar` file. The Server will unpack the archive and store each file individually, as if they were uploaded one-by-one.

The Editor page for the Unnamed Skin contains the list of all Named Skins:

| Named Skins |
| :---: |
| GoldenFleece |
| IceColdMail |
| |

To create a Named Skin, enter its name, and click the Create button.

To remove Named Skins, use the checkboxes to select the Skins, and then click Remove Marked button. Only empty Skins (Skins without any files) can be removed.

To remove the Unnamed Skin, remove all its files and all Named Skins, and then click Remove Unnamed Skin button.

To open a Skin, click its name. The Editor will display the Skin Name, and it will provide the UP link to the Unnamed Skin page.

The Named Skin Editor allows you to rename the Skin by entering a new Skin Name and clicking the Rename Skin button.

## CLI/API

The Command Line Interface/API can be used to manage Skins. See the [Web Skins Administration](#) section for the details.

## Virtual File Storage Area

The Skins can be modified using [FTP](#) and [HTTP](#) clients, [CG/PL](#) applications, and any other method that provides access to the Account [File Storage](#).
Special `$DomainSkins`, `$ServerSkins`, and `$ClusterSkins` directories provide access to the Domain and Server/Cluster-wide Skins.

---

# Request Processing

The CommuniGate Pro Web Application module processes a request for a WSSP file by calling a code component that produces a *dataset* - a [dictionary](#) containing text string keys and values, associated with those keys. Values can be text strings, arrays of values, or dictionaries.

For example, when a Domain default page is requested, the code component called and it produces a dictionary that contains keys such as `canAutoSignup`, `hasMailLists`, `hasCertificate`.

The Web Application module then uses the script code from a WSSP file to convert this data into into an HTML or other markup language page.

This section lists the available CommuniGate Pro code components, specifies when those components are called, explains how the code components process the <FORM> parameters, and specifies the content of the dataset produced by each code component.

---

# Code Components for Stateless Requests

The Web Application module checks the `Skin` HTTP parameter for all Stateless requests. If this parameter exists, the module tries to open a Named Skin with the specified name, otherwise the Unnamed Skin (for the addressed Domain) is used.

The Web Application module checks the `Language` HTTP parameter for all Stateless requests. If this parameter exists, the module uses it to selected a non-default Text DataSet for the selected Skin.

The Web Application module places certain data into datasets produced with all Stateless Requests code components. The following list specifies these "generic" dataset elements that can be used with all Stateless WSSP pages:

filesRef
>This element value is a string that can be used to form a URL that refers to a file (image, style sheet, data, etc.) from the same Skin in the same Domain. The `<img src="%%filesRef%%Logo.jpeg">` HTML element will display the `Logo.jpg` file "visible" in the current Skin.

serverName
>This element value is a string with this CommuniGate Pro server name (its Main Domain Name).

skinName
>This optional element value is the current Skin name string.

language
>If a non-default language was selected, this optional element contains a string - the selected language name.

domainName
>This element value is a string containing the CommuniGate Pro Domain name. This element exists only if the Server has succeeded to direct the Stateless Request to one of the server Domains.

charset
>This element value is the value of the `charset` element from the Domain Skin Text Dataset. Individual code components may specify other values for the `charset` element (see below).

secureChannel
>This element exists and has the `YES` string value if the request has been received via a secure (HTTPS) connection.

isWML
>This element exists and has the `YES` string value if the request has been received from a WML browser.

The following sections specify the Stateless URLs, the name of the code component called, the actions taken by the component, the dataset produced with that component, and the name of the WSSP file used to produce the HTTP response.

---

**URLs: `/,/default.html`**

these URLs are used to process Login operations.

**Actions**

>If the HTTP request has the `username` and `password` parameters, the code tries to authenticate the client using these parameter values. If the supplied credentials are correct, a new WebUser Session is created, and a request for the "entry page" is sent to the Session. This request usually returns an HTML "jump page" that is sent back to the user browser. It forces the browser to enter the "Session realm".

>If the request has the `DisableIPWatch` parameter, the "Fixed IP Address" security feature will be disabled for this session, even if the Account WebUser preferences enable it.

>If the request has the `DisableUseCookie` parameter, the "Use Cookies" security feature will be disabled for this session, even if the Account WebUser preferences enable it.

>If the request has `SessionSkin` parameter with a string value not equal to `*`, the session is opened using the Skin specified with this parameter. The Skin is searched in the Domain of the logged-in user (which can be different than the Domain used

to display the Login page).

## Result Dataset

If `username` or `password` parameter has not been specified, or a WebUser Session could not be created, the component generates the following *dataset*:

autoSignup
> this element (with the string `YES` as the value) is added to the dataset if the addressed Domain provides the Auto-Signup operation.

clientAddress
> this string element contains the IP address of the user browser.

errorCode
> this element is added to the dataset if the Login operation has failed. The value is a string with the error code.

forgotPassword
> this element (with the string `YES` as the value) is added to the dataset if the `errorCode` value is "Incorrect Password" or "Unknown Account".

hasCertificate
> this element (with the string `YES` as the value) is added to the dataset if the addressed Domain has a custom [Certificate](Certificate).

hasDirectory
> this element (with the string `YES` as the value) is added to the dataset if the default Skin for the addressed Domain has an array element `GuestDirectoryFields` in its Text Dataset, and that array is not empty.

hasLists
> this element (with the string `YES` as the value) is added to the dataset if the addressed Domain has at least one Mailing List. This element is always added if the addressed Domain is a Shared Domain.

skinNames
> this element contains an array - the list of skin names available in the addressed Domain.

loginName
> this string element is added to the dataset if the user has tried to log in, but failed. The value specified in the username parameter becomes the loginName element value.

restoreSessionPage
> this string element is added to the dataset if the user has been disconnected. It contains the name of the interrupted request session page name.
> To allow an interrupted request to resume, the `restoreSessionPage` HTTP parameter with this element value should be included into the HTTP request generated by this page.

restoreCharset
> this string element is added to the dataset if the user has been disconnected. It contains the charset used in the interrupted request.
> To allow an interrupted request to resume, the `restoreCharset` HTTP parameter with this element value should be included into the HTTP request generated by this page.

restoreParameters
> this array element is added to the dataset if the user has been disconnected. Each array element is a dictionary with the following elements:

> name
> > name of an interrupted request parameter.

> value
> > encoded value of an interrupted request parameter.

> To allow an interrupted request to resume, parameters with these names and values should be included into the HTTP request generated by this page.

## WSSP page

the `login.wssp` page is used to generate the HTTP response.


If login operation was successful, the HTTP Redirect response is returned, with the Redirect URL pointing to the *StartPage* wssp page within a newly created Session. The StartPage is specified as the `StartPage` Skin string.

If login operation was successful, but the request contained the `restoreSessionPage` parameter, the `resume.wssp` page is displayed (as a Stateless one).

The Result Dataset for this page contains:

`restoreParameters`
  see description above.

`sessionID`
  the SessionID for the newly created session (the same value as the value of `SESSION(ID)` function that can be used on session wssp pages).

`jumpPage`
  the wssp page to open if the user wants to resume the interrupted operation (it includes optional parameters).

---

## URL: `/RecoveryPassword.wssp`

this URL is used to process Password Recovery operations.

### Actions

If the HTTP request has the `username` and `Send` parameters, the component tries to find the specified Account, retrieves the Account password, and the `RecoverPassword` Account Setting. If the password can be decrypted and the RecoverPassword is specified, an E-mail message containing the password is composed and sent to the RecoverPassword E-mail address.

### Result Dataset

`errorCode`
  this element is added to the dataset if the Password Recovery operation has failed. The value is a string with the error code.

`messageCode`
  this element is added to the dataset if the Password Recovery operation has succeeded. The value is the `PasswordSent` string.


### WSSP page

the `recoverypassword.wssp` page is used to generate the HTTP response.

---

## URL: `/Signup.wssp`

this URL is used to process Auto-Signup operations.

### Actions

If the HTTP request has the `username`, `password1`, `password2`, and the `realName` parameters, the component tries to create the specified Account. Before it tries to create an Account, it checks if the `password1` and `password2` strings are the same. If a new Account is created, its `UseAppPassword` setting is set to `YES`, the `Password` and `RealName` settings are set to the specified values. If the HTTP request contains a non-empty ForgotPassword parameter, it is used as the RecoverPassword Account Setting.

The component checks if the request contains one or more `PublicInfo` string parameters. The value of the parameter must be one of the Public Info Attributes specified in the [Directory Integration](#) settings. The component than checks if there is a non-empty request parameter with this name, and adds the parameter value to the initial Account settings.

Example: to provide a `City` field on the Auto-Signup page, include the `<INPUT type="hidden" name="PublicInfo" value="City">` control and the `<INPUT type="text" name="City" value="" size=30 maxlength=255>` control into the Signup.wssp HTML code.

If an Account has been created, a new WebUser Session is created, and a request for the "entry page" is sent to the Session (see above).

## Result Dataset

If `username`, `password1`, `password2`, or the `realName` parameter has not been specified in the HTTP request, or a new Account could not be created, the component generates the following *dataset*:

`errorCode`
> this element is added to the dataset if the Auto-Signup operation has failed. The value is a string with the error code.

`userName`
> this element is added to the dataset if HTTP request contained a non-empty `userName` parameter. The element value is the value of the parameter.

`realName`
> this element is added to the dataset if HTTP request contained a non-empty `realName` parameter. The element value is the value of the parameter.

`recoverPassword`
> this element is added to the dataset if HTTP request contained a non-empty `recoverPassword` parameter. The element value is the value of the parameter.

## WSSP page

the `signup.wssp` page is used to generate the HTTP response.

---

## URL: `/List/`, `/List/default.html`

this URL is used to retrieve the list of the browsable Domain [Mailing Lists](#).

### Actions

The HTTP request parameters are not processed.

### Result Dataset

The component generates the following *dataset*:

`errorCode`
> this element is added to the dataset if the list retrieval operation has failed. The value is a string with the error code.

`lists`
> this element contains an array of mailing list descriptors. Each descriptor is a dictionary with the following keys and values:

> `name`
>> a string with the mailing list name.

> `realName`
>> the mailing list "description" string.

> `browse`
>> the string describing the mailing list archive browsing policies. The string value can be `anyone` or `subscribers`.

### WSSP page

the `listlist.wssp` page is used to generate the HTTP response.

**URL:** `/List/`*`listname`*`/`, `/List/`*`listname`*`/List.html`

this URL is used to retrieve a portion of the mailing list records for the *listname* [Mailing List](#).

The code component actually uses the generic [Mailbox](#) Code Component.

**Actions**

The component checks if the HTTP request contains the `NextMessage` parameter with a numeric value. If it exists, the value is interpreted as the unique message ID (UID) in the list archive Mailbox, and the component tries to find this message in the selected Mailbox "view", and tries to find the next message in the view.

The component checks if the HTTP request contains the `PrevMessage` parameter with a numeric value. If it exists, the value is interpreted as the unique message ID in the list archive Mailbox, and the component tries to find this message in the selected Mailbox "view", and tries to find the previous message in the view.

If the next or previous message is found, its UID is added to the dataset (see below) and the generic Mailbox component is not used for processing.

If no next/previous message is found, the generic Mailbox component is used to process the HTTP request parameters and to compose the resulting dataset.

**Result Dataset**

`listName`
a string with the Mailing List name.


If a next or previous message is found:

`messageJump`
a string with the found message UID.


If a next or previous message was not requested in the HTTP request parameters or it was not found:

`realName`
the Mailing List Description string

`charset`
the Mailing List Preferred charset string


the generic Mailbox code component is used to generate the rest of the resulting dataset.


**WSSP page**
the `listmailbox.wssp` page is used to generate the HTTP response.

---

**URL:** `/List/`*`listname`*`/Message/`*`uid`*`.html`

this URL is used to retrieve the message with *uid* unique ID from the *listname* mailing list archive.

The code component actually uses the generic [Message](#) Code Component.

**Actions**
The generic Message component is used to process the HTTP request parameters and to compose the resulting dataset.

**Result Dataset**

listName
>   a string with the Mailing List name.

nextMsg
>   this element is added if there is a next message in the archive Mailbox view. The element value is a string with the next message UID.

prevMsg
>   this element is added if there is a previous message in the archive Mailbox view. The element value is a string with the previous message UID.

The generic Message code component is used to generate the rest of the resulting dataset.

**WSSP page**
>   the `listmessage.wssp` page is used to generate the HTTP response.

---

# Error Pages

The WSSP mechanism is used to generate HTTP response body for error responses. The following table lists the HTTP error codes, the situations when this error code is generated, and the WSSP file used to product the HTTP error response body.

| Code | Error conditions | WSSP file used |
|------|------------------|----------------|
| 301 | the requested resource has been moved | moved.wssp |
| 404 | the requested resource does not exist | notfound.wssp |
| 401 | the requested page requires the HTTP Authorization (request header) | unauthorized.wssp |
| 401 | the credentials in the HTTP Authorization request header are incorrect | denied.wssp |
| 500 | generic system error | failure.wssp |
| 501 | generic system error | error.wssp |

These WSSP pages are processed using datasets generated for all Stateless requests, with the following additional elements:

errorCode
>   this element is added to the dataset if there is an error code to be reported to the user.

hostName
>   this element is added to the dataset if the HTTP request contained the `Host:` field. The element value is that request field parameter.

The `disconnected.wssp` page is used when an HTTP request was sent to a WebUser Session, but the session has not been found. The page is processed using a dataset generated for all Stateless requests.

---

# Code Components for Session Requests

When a new WebUser Session is created, the Skin specified in the Account WebUser Preferences is opened and is used as the "Session Skin". The string `StartPage` is retrieved from that Skin Text Dataset, and a jump page is composed and sent to the user browser. The jump page redirects the user browser into the "Session Realm", to the page specified with the `StartPage` string.

HTTP requests to the "Session realm" (requests with URLs started with `/Session/`) are processed as Session Requests. The second component of the Session Request URL is a unique Session ID, the HTTP module uses it to find the WebUser session.

If the specified session is not found, or the Session has the `Fixed Network Address` option set, and the HTTP request did not come from the same IP address as the Login request that started the session, the `disconnected.wssp` page is displayed (see above).

After the Session is found, the Web Application module processes the rest of the request URL as the "request inside this session realm". If the request URL specifies a regular file, that file is retrieved from the Session Skin, and it is sent back to the user browser.

For each `.wssp` request a code component is called. It processes the HTTP request parameters and generates a *result dataset*. Then the .wssp file is retrieved from the Skin, and it is used to compose the HTTP response.

If the result dataset does not contain the `blockAlerts` element, the Web Application module checks if there is a pending [Alert](#) for the session user. If one or several pending alerts are found, the `Alerts` code component is called, and the `Alerts.wssp` file is used to compose the HTTP response.

The Web Application module checks for certain HTTP request parameters and processes them for all .wssp page requests. The following list specifies these "generic" actions:

`EmptyTrashNow`
> If the HTTP request contains this parameter, and the Mailbox or Mailbox alias with the name `Trash` can be opened with the "Can Delete" Mailbox Access Right, then the code component removes all messages from that Mailbox. If this operation fails, the error code is placed into the resulting dataset as the `errorCode` string element.

`SMIMEUnlock`
> If the HTTP request contains this parameter and the session does not have an Active Private Key, and the encrypted Private Key exists in the Account Settings, and the HTTP request contains the `SMIMEPassword` parameter, the module tries to activate ("unlock") the Private Key using the supplied password. If the operation fails, the error code is placed into the resulting dataset as the `SMIMEError` string element.

The Web Application module places certain data into datasets produced with all Session Requests code components. The following list specifies these "generic" dataset elements that can be used with all Session WSSP pages:

`messageText`
> This string element is added if the HTTP request contains the `messageText` parameter. The element value is the same as the HTTP parameter value.

`messageCode`
> This string element is added if the HTTP request contains the `messageCode` parameter. The element value is the same as the HTTP parameter value.

`secureChannel`
> This element exists and has the `YES` string value if the request has been received via a secure (HTTPS) connection.

`isWML`
> This element exists and has the `YES` string value if the request has been received from a WML browser.

`charset`
> This string element contains the Preferred Charset selected in the User Preferences, if the Use UTF8 mode is set to Never. Otherwise, this element contains the `utf-8` string.
> **Note:** it's just a default value, individual code components can specify different values for this dataset element.

`SMIMEActive`
> This element exists and has the `YES` string value if the session has an unlocked (Active) SMIME Private Key.

`SMIMEInactive`
> This element exists and has the `YES` string value if the session does not have an unlocked (Active) SMIME Private Key, but the user has the Private Key in the Account Settings, and the Key can be unlocked.

`mailboxes`
> The list of all selectable Mailboxes visible to the user.

If a .wssp request specifies an unknown code component, but the .wssp file with the specified name can be retrieved from the Session Skin, that .wssp file is processed using the dataset with the "generic" elements only, and the result is sent back to the user browser.

<div align="center">Hello.wssp</div>

Example: The Stock Skin uses ⬚ requests. There is no code component with this name, so a dataset with the generic values is composed, and the `Hello.wssp` file is used to process this dataset.

The following sections specify the names of existing code components (names for .wssp requests), the actions taken by these component, and the dataset produced with these components.

The code component results are processed using the .wssp files with the same names as the code component names.

---

**Name: `Mailboxes`**

**Actions**

If the HTTP request contains the `Create` parameter and the `NewName` parameter is a non-empty string, the component tries to create a Mailbox with the specified name. If this operation fails, the `errorCode` element with the error code text is added to the result dataset. If the Mailbox is created, the `messageCode` element with `MailboxCreated` string value is added to the result dataset, and the created Mailbox name is added to the list of subscribed Mailboxes, if the Show Subscribed Mailboxes option is selected in the Account WebUser Preferences.
If the request contains the `newClass` parameter, then the created Mailbox is set to the specified class.

If the HTTP request contains the `Filter` parameter, only the Mailboxes with names containing this parameter value are included into the list.

**Result Dataset**

The code component creates a list of all Account Mailboxes and Mailbox Aliases (if the Show All Account Mailboxes option is selected in the Account WebUser Preferences), or the list of all subscribed Mailboxes (if the Show Subscribed Mailboxes option is selected). If both options are selected, these two lists are merged into one.

filter
> this string element contains the current value of the HTTP `Filter` parameter.

newName
> this string element contains the current value of the HTTP `NewName` parameter.

mailboxClasses
> this array elements contains strings - names of all supported Mailbox classes.

mailboxList
> this element is an array with one dictionary-type element for each Mailbox in the composed Mailbox list. Each dictionary contains the following elements:
>
> mailboxName
> > this string element contains the Mailbox name.
>
> parent
> > this string element exists if the Mailbox is a submailbox of some other Mailbox. The string contains the name of that parent Mailbox.
>
> nonSelectable
> > this string element with the value `Yes` is added if the Mailbox is not selectable. If it is added, none of the following elements is added to the dictionary.
>
> isList
> > this element is added if the Mailbox is the main Mailbox (archive) for a Mailing List (this also means that there is a Mailing List with the same name).
>
> nMessages
> > this string element contains the number of messages in the Mailbox. If the number cannot be retrieved, the element value is the `???` string.
>
> nRecent
> > this string element contains the number of "Recent" messages in the Mailbox.

1078

numUnread
> this string element contains the number of "Unseen" messages in the Mailbox.

size
> this string element contains the "rounded" size of the Mailbox.

mailboxClass
> this optional string element contains the Mailbox class (for non-mail Mailboxes).

mailboxPage
> this string element contains the name of the wssp page to be used to process this Mailbox class.


nSelected
> this string element contains the number of elements in the `mailboxList` array.

trashSize
> this string element is added only if the Account has the `Trash` Mailbox. It contains the `Trash` Mailbox size.

currentStorage
> this string element contains the "rounded-up" total size of Account Mailboxes.

storageLimit
> this string element contains the "rounded-up" Account total Mailbox size limit. If the Account does not have the total Mailbox size limit, this element contains the `unlimited` string.

---

### Name: `Mailbox`

The HTTP request must contain the `Mailbox` parameter - the name of the Mailbox to be displayed.

### Actions

For each Mailbox, the module creates a session object that contains the Mailbox view parameters. When the object is created, these parameters are initiated with the WebUser Preferences values.

The HTTP request `Msg` parameters are interpreted as "message set elements". A request can contain several parameters, and each parameter should have a numeric value - the Unique ID of a Mailbox message.

If the HTTP request contains the `Forward` or `Redirect` parameter and the `RedirectAddresses` parameter is not empty, a "message set" is composed using the `Msg` parameters, and the message set messages are forwarded or redirected to the specified addresses. If the HTTP request contains the `ListApprove` parameter and the Mailbox is an "approval" or "requests" Mailbox for some mailing list, the request is processed as the `Redirect` request with the effective address being the mailing list address or the "subscription" request for that list.
If the operation has been successful, the `messageCode` element with the `MessagesForwardedInfo` or `MessagesRedirectedInfo` string value is added to the result dataset. Otherwise, the `errorCode` element with the operation error code string value is added to the result dataset.

If the HTTP request contains the `Copy` or `Move` parameter and the `MailboxName` parameter contains a name of some selectable Mailbox, a "message set" is composed using the `Msg` parameters, and the message set messages are copied to the specified Mailbox. If the `Move` parameter was specified, the message set messages are marked as Deleted, physically deleted, or moved to Trash - depending on the WebUser Preferences.
If the operation has been successful, the `messageCode` element with the `MessagesCopiedInfo` string value is added to the result dataset. Otherwise, the `errorCode` element with the operation error code string value is added to the result dataset.

If the WebUser Preferences DeleteMethod option is set to `Move To Trash`, and the HTTP request contains the `Delete` parameter, a "message set" is composed using the `Msg` parameters, the message set messages are copied to the Trash Mailbox and deleted. If the `Trash` Mailbox did not exist, it is created.

If the HTTP request contains the `DeleteAll` parameter, all Mailbox messages are deleted, using the method specified with the WebUser Preferences DeleteMethod option.

If the HTTP request contains the `read`, `unread`, `flag`, `unflag`, `delete`, or `undelete` parameters, a "message set" is composed using the `Msg` parameters, and the flags for the message set messages are modified. The `delete` and `undelete` parameters are processed in this way only if the WebUser Preferences DeleteMethod option is not set to `Move To Trash`.
If the operation has not been successful, the `errorCode` element with the operation error code string value is added to the result dataset.

If the WebUser Preferences DeleteMethod option is not set to `Move To Trash`, and the HTTP request contains the `Purge` parameter, all Mailbox messages with the `Deleted` flag are deleted.
If the operation has not been successful, the `errorCode` element with the operation error code string value is added to the result dataset.

If the HTTP request contains the `NextMessage` parameter with a numeric value, the value is interpreted as the unique ID (UID) of a Mailbox message, and the component tries to find the next Mailbox message. If such a message is found, its UID is added to the Result Dataset as the `messageJump` element.

If the HTTP request contains the `PrevMessage` parameter with a numeric value, the value is interpreted as the UID of a Mailbox message, and the component tries to find the previous Mailbox message. If such a message is found, its UID is added to the Result Dataset as the `messageJump` element.

If the HTTP request contains the `NextUnread` parameter and the Mailbox contains an unread message, the UID of that unread message is added to the Result Dataset as the `messageJump` element.

If the messageJump element was not added to the Result Dataset, the code component uses the generic Mailbox component to process the HTTP request parameters and to compose the resulting dataset.

**Result Dataset**

`mailbox`
    a string with the Mailbox name.

`mailboxClass`
    if this string element exists, it contains the Mailbox class.

`mailboxPage`
    this string element contains the name of the wssp page that should be used to display Mailboxes of this class.

`isSentBox`
    this element exists and contains the YES string if the current Mailbox is the Mailbox selected to keep copies of sent messages.

`isDraftsBox`
    this element exists and contains the YES string if the current Mailbox is the Mailbox selected to keep message drafts.


If a next unread, next, or previous message is found:

`messageJump`
    a string with the found message UID.


If a next unread, next, or previous message was not requested in the HTTP request parameters or it was not found:

`refreshTime`
    the Mailbox view refresh time (in seconds), retrieved from the WebUser Preferences.

`listApproval`
    this string element exists if the Mailbox is the *approval* Mailbox for a mailing list. The element contains the E-mail address of that mailing list.


The generic Mailbox code component is used to generate the rest of the resulting dataset.

---

**Name: `Contacts`**

Processed in the same way as the Mailbox page.

---

### Name: `Notes`

Processed in the same way as the Mailbox page.

---

### Name: `Calendar`

The HTTP request must contain the `Mailbox` parameter - the name of a Calendar-type Mailbox to be displayed.

**Actions**

For each Mailbox, the module creates a session object that contains the Mailbox view parameters. When the object is created, these parameters are initiated with the WebUser Preferences values. The object also contains the month number for the "monthly calendar" view. It is initially set to the current month. The object contains the day number that specifies the first day to be displayed in the Calendar view. The object also contains the "byDay" flag that controls how the calendar data is stored in the dataset (by days or by time intervals).

The HTTP request `prevMonthlyCalendar` parameter can specify the number of months to be subtracted from the "monthly calendar" month number.

The HTTP request `nextMonthlyCalendar` parameter can specify the number of months to be added to the "monthly calendar" month number.

The HTTP request `JumpDay` parameter can specify the "day number in the epoch" that will become the first day to be displayed in the Calendar view.

The HTTP request `byDay` parameter can specify the new byDay flag value.

The HTTP request `Msg` parameters are interpreted as "message set elements". A request can contain several parameters, and each parameter should have a numeric value - the Unique ID of a Mailbox message.

If the WebUser Preferences DeleteMethod option is set to `Move To Trash`, and the HTTP request contains the `Delete` parameter, a "message set" is composed using the `Msg` parameters, the message set messages are copied to the Trash Mailbox and deleted. If the `Trash` Mailbox did not exist, it is created.

If the HTTP request contains the `read`, `unread`, `flag`, `unflag`, `delete`, or `undelete` parameters, a "message set" is composed using the `Msg` parameters, and the flags for the message set messages are modified. The `delete` and `undelete` parameters are processed in this way only if the WebUser Preferences DeleteMethod option is not set to `Move To Trash`.
If the operation has not been successful, the `errorCode` element with the operation error code string value is added to the result dataset.

If the WebUser Preferences DeleteMethod option is not set to `Move To Trash`, and the HTTP request contains the `Purge` parameter, all Mailbox messages with the `Deleted` flag are deleted.
If the operation has not been successful, the `errorCode` element with the operation error code string value is added to the result dataset.

**Result Dataset**

`mailbox`
     a string with the Mailbox name.

`refreshTime`
     the Mailbox view refresh time (in seconds), retrieved from the WebUser Preferences.

`weekDayNames`
     the array containing weekday name strings, starting with the day specified as the first weekday in the WebUser Preferences.

`todayDay`

the day of month for the current date.

todayMonth
>    the current month.

todayYear
>    the current year.

todayDayNum
>    the "day number in the epoch" for the current day.

monthlyCalendar
>    an array containing one element for each week of the months to be displayed in the "monthly calendar". Each week element is an array with 7 elements. If the element does not correspond to a month day (i.e. the element corresponds to a day before the first day of month or after the last day of month), the element is an empty string. Otherwise the element is a dictionary containing the following subelements:
>
>    day
>>    a string with the day of the month corresponding to this element.
>
>    workDay
>>    an optional YES string added if the day is a working day.
>
>    dayNum
>>    the "day number in the epoch" for this day.

year
>    the string with the number of the year the first displayed day belongs to.

byDay
>    the optional element containing the YES string. It is added if the byDay flag is set.

timeSlices
>    an array containing time slice starting times if the byDay flag is set. Each element is a dictionary containing the following values:
>
>    hour
>>    the hour number in the 24-hour system.
>
>    PMhour
>>    the hour number in the 12-hour system if the hour value is 12 or more.
>
>    minute
>>    the minute number. Always 2 digits.

calendarDays
>    an array containing calendar view elements if the byDay flag is set. Each element is a dictionary presenting calendar data for one day. It contains the following elements:
>
>    weekDay
>>    the weekday name of this day
>
>    year
>>    the number of the year this day belongs to
>
>    month
>>    the name of month this day belongs to
>
>    day
>>    the day number in the month
>
>    dayNum
>>    the "day number in the epoch"

**allDayEvents**

an optional array containing descriptors for all-day events for this day. Each array element is a dictionary containing "Event elements" (see below)

**events**

an array containing descriptors this day events. Each descriptor corresponds to one time interval, it is a dictionary containing "Event elements" (if there is an Event in this time interval) and the following elements:

**nTimeSlices**

the length of the descriptor time interval, in time slices.

**conflicts**

an optional array containing UIDs of other Events conflicting with the Event displayed in this time interval.

**status**

if the time interval does not contain an Event and it does not belong to a "working time" period, this element contains the UNAVAILABLE string.

**calendarDays**

an array containing calendar days if the byDay flag is not set. Each element is a dictionary containing the following values:

**weekDay**

the weekday name of this day

**year**

the number of the year this day belongs to

**month**

the name of month this day belongs to

**day**

the day number in the month

**dayNum**

the "day number in the epoch"

**allDayEvents**

an array containing information about All-Day Events if the byDay flag is not set. The array has one element for each displayed day. This element is an empty string if there is no All-Day Events in that day, or it is an array containing dictionary subelements for each All-Day Event taking place that day. Each dictionary subelement contains the "Event elements" for one All-Day Event.

**calendarSlices**

an array containing information for a time interval if the byDay flag is not set. Each array element is a dictionary containing the following elements:

**hour**

the hour number in the 24-hour system.

**PMhour**

the hour number in the 12-hour system if the hour value is 12 or more.

**minute**

the minute number. Always 2 digits.

**days**

an array with the calendar data for this time interval in each day. Each element is a dictionary with the day data containing the "Event elements" if this time interval contains an Event in that day, and also the following elements:

**nTimeSlices**

the length of the descriptor time interval, in time slices.

conflicts
> an optional array containing UIDs of other Events conflicting with the Event displayed in this time interval.

status
> if the time interval does not contain an Event and it does not belong to a "working time" period, this element contains the `UNAVAILABLE` string.

The "Event elements" are:

summary
> the string with Event Summary text.

ID
> a numeric string with the UID of the Event message in the Mailbox.

status
> a string with the Event busy status.

priority
> a numeric with the Event priority value. This element exists only if the Event priority is not zero.

---

## Name: `Tasks`

The HTTP request must contain the `Mailbox` parameter - the name of a Tasks-type Mailbox to be displayed.

### Actions

For each Mailbox, the module creates a session object that contains the Mailbox view parameters. When the object is created, these parameters are initiated with the WebUser Preferences values. The object contains the day number that specifies the first day to be displayed in the Tasks view.

The HTTP request `JumpDay` parameter can specify the "day number in the epoch" that will become the first day to be displayed in the Tasks view.

The HTTP request `Msg` parameters are interpreted as "message set elements". A request can contain several parameters, and each parameter should have a numeric value - the Unique ID of a Mailbox message.

If the WebUser Preferences DeleteMethod option is set to `Move To Trash`, and the HTTP request contains the `Delete` parameter, a "message set" is composed using the `Msg` parameters, the message set messages are copied to the Trash Mailbox and deleted. If the `Trash` Mailbox did not exist, it is created.

If the HTTP request contains the `read`, `unread`, `flag`, `unflag`, `delete`, or `undelete` parameters, a "message set" is composed using the `Msg` parameters, and the flags for the message set messages are modified. The `delete` and `undelete` parameters are processed in this way only if the WebUser Preferences DeleteMethod option is not set to `Move To Trash`.
If the operation has not been successful, the `errorCode` element with the operation error code string value is added to the result dataset.

If the WebUser Preferences DeleteMethod option is not set to `Move To Trash`, and the HTTP request contains the `Purge` parameter, all Mailbox messages with the `Deleted` flag are deleted.
If the operation has not been successful, the `errorCode` element with the operation error code string value is added to the result dataset.

The HTTP request `showCompleted` parameter can specify the new showCompleted flag value.

### Result Dataset

mailbox
> a string with the Mailbox name.

refreshTime

the Mailbox view refresh time (in seconds), retrieved from the WebUser Preferences.

showCompleted
> the optional element containing the YES string. It is added if the showCompleted flag is set.

numTotal
> the total number of calendaring items in the Mailbox.

numSelected
> the total number of selected Task items.

tasks
> an array with selected tasks. Each element is a dictionary describing a task. It contains the following elements:

> nBefore
>> this number string exists if the Task starts after the initial time displayed in the Tasks view. It shows how many Task view time periods should be skipped before the Task starts.

> nDuration
>> this number string specifies the time interval (in time periods) between either the Task start time or the Task view first display time (whatever is later) and the Task Due time or the Task view last display time (whatever is earlier).

> nAfter
>> this numeric string exists if the Task ends before the Task view end time. It shows how many Task view time periods should be skipped after the Task ends.

> ID
>> a string with the UID of the Task message in the Mailbox.

> percentComplete
>> a numeric string with the Percent-Complete value of the Task object.

> summary
>> a string with the Task summary

> priority
>> a numeric string with the Task priority if it was set (is not zero).

---

## Name: `Message`

The HTTP request must contain the `Mailbox` parameter (the name of the Mailbox containing the messages to be displayed), and the `MSG` parameter - the Unique ID of that message in the Mailbox.

**Actions**
> If the HTTP request contains the `Copy` or `Move` parameter and the `MailboxName` parameter contains a name of some selectable Mailbox, the message is copied to the specified Mailbox. If the `Move` parameter was specified, the message is marked as Deleted, or it is deleted - depending on the WebUser Preferences.
> If the operation has been successful, the `messageCode` element with the `MessageCopied` string value is added to the result dataset. Otherwise, the `errorCode` element with the operation error code string value is added to the result dataset.
> If the Move operation has removed the message, the `backToMailbox` element with the `Yes` value is added to the result dataset, and the code component stops request processing.

> If the HTTP request contains the `Redirect` parameter and the `RedirectAddresses` parameter is not empty,the message is redirected to the specified addresses.
> If the HTTP request contains the `ListApprove` parameter and the message Mailbox is an "approval" Mailbox for some mailing list, the request is processed as the `Redirect` request with the effective address being the mailing list address.
> If the operation has been successful, the `messageCode` element with the `MessageRedirected` string value is added to the result dataset. Otherwise, the `errorCode` element with the operation error code string value is added to the result dataset.

> If the HTTP request contains the `TakeAddress` parameter the message `From:` address is added to the Account address book.

If the HTTP request contains the `TakeCertificate` parameter the certificate from the message digital signature is added to the Account address book.

If the HTTP request contains the `StoreFiles` parameter and the `selectedWebFolder` parameter contains a name of the File Storage folder, the file parts of the message (attachments, images) are stored in the specified folder.
If the operation has been successful, the `messageCode` element with the `FilesCopied` string value is added to the result dataset. Otherwise, the `errorCode` element with the operation error code string value is added to the result dataset.

If the HTTP request contains the `read`, `unread`, `flag`, `unflag`, `delete`, or `undelete` parameters, the message flags are modified.
If the operation has not been successful, the `errorCode` element with the operation error code string value is added to the result dataset.

Then the code component use the generic Message component to process the HTTP request parameters and the compose the resulting dataset.

**Result Dataset**

mailbox
> A string with the Mailbox name.

If the message has not been removed:

MSG
> A string with the message UID.

flagged, recent, deleted, flagged, media, isDraft
> These elements with the `Yes` value are added if the message has the corresponding flags.

status
> This string element has the following values:

> - `Deleted` - if the message has the Deleted flag set, otherwise
> - `Draft` - if the message has the Draft flag set, otherwise
> - `Redirected` - if the message has the Redirected flag set, otherwise
> - `Unread` - if the message does not have the Seen flag set, otherwise
> - `Answered` - if the message has the Answered flag set, otherwise
> - `Read`

messageBody
> A string with HTML presentation of the message, generated using the generic Message code component.

charset
> The charset to use for message display. This element can be set with the generic Message code component.

nextMsg
> this element is added if there is a next message in the Mailbox view. The element value is a string with the next message UID.

prevMsg
> this element is added if there is a previous message in the Mailbox view. The element value is a string with the previous message UID.

hasFiles
> this `YES` string element is added if there is the message contains a file part.

editableContact
> this `YES` string element is added if the message is a VCard object that can be updated.

editableGroup
> this `YES` string element is added if the message is a Group object that can be updated.

editableNote
> this YES string element is added if the message is a Note object that can be updated.

editableEvent
> this YES string element is added if the message is an Event "published" by this user and it that can be updated.

editableTask
> this YES string element is added if the message is a Task "published" by this user and it that can be updated.

canCancelEvent
> this YES string element is added if the message is an Event this user can cancel.

canCancelTask
> this YES string element is added if the message is a Task this user can cancel.

canAcceptDecline
> this YES string element is added if the message is a Task or Event request.

percentComplete
> this element containing a number is added if the message is an Task delegated to this user by someone else.

statusCode
> this optional string element contains the status of the message if the message is a Task or an Event.

conflictingID
> this optional string element contains the UID of the Default Calendar Mailbox message that conflicts with the displayed Event Request.

canUpdatePartStatus
> this YES string element is added if the message is a reply to the user's Task or Event request.

canCancelEvent
> this YES string element is added if the message is a cancel request from an Event organizer.

canCancelTask
> this YES string element is added if the message is a cancel request from a Task organizer.

listApproval
> this string element exists if the message Mailbox is the *approval* Mailbox for a mailing list. The element contains the E-mail address of that mailing list.

---

## Name: Compose

### Actions

If the HTTP request contains the charset parameter, the parameter value is used as the *desired* charset - the charset to be used in the composed message.

The optional Operation HTTP request parameter specifies the type of the Compose operation and it can have the Reply, ReplyAll, Forward, or EditDraft value.
If this parameter is specified, the OrigMessage parameter (with the UID of the original message) and the OrigMailbox parameter (with the name of the Mailbox containing the original message) must be specified.

If the HTTP request contains the Operation parameter and it does not contain the filled parameter, the original message header fields are used to compose the Subject, To, Cc, and the message body data for the new message.
Otherwise, the Subject, To, Cc, Bcc, and Body HTTP request parameters are used as the new message data.

If the HTTP request contains the AddressBook parameter and it does not contain the CloseBook parameter, or if HTTP request contains the OpenBook parameter the generic AddressBook code component is used to process the request parameters and to form some result dataset elements.

If the HTTP request contains the isEvent parameter, the Calendar Event item is being composed. If the HTTP request

contains the `isTask` parameter, the Calendar Task (ToDo) item is being composed. If the HTTP request contains the `isNote` parameter, the Note item is being composed.

If the HTTP request contains the `Send` parameter, the composed message is submitted to the Server Queue. If the HTTP request contains the `Save` parameter, the composed message is stored as a Draft in the selected Drafts Mailbox.
In both cases all HTTP request `Attachment` parameters are added to the message as attachments.

## Result Dataset

operation
> This string element is added to the result dataset if the HTTP request contains the `Operation` parameter. The element value equals the request parameter value.

origMessage
> This element containing the UID of the original message is added to the result dataset if the HTTP request contains the `OrigMessage` parameter.

origMailbox
> This element with the name of the Mailbox containing the original message is added to the result dataset if the HTTP request contains the `OrigMailbox` parameter.

sentOrSaved
> This element with `Yes` value is added to the result dataset if the Send or SaveDraft operation has completed successfully. If this element is added:

> * The `sent` element with the `Yes` value is added if the operation was the Send operation.
> * The `messageCode` element with the `MessageSent` or `MessageSaved` value is added to dataset.
> * No other element listed below is added to the result dataset.

Subject, To, Cc, Bcc
> These elements contain strings with the current header fields data.

From
> This element contains a string with the `From` address specified in the WebUser Preferences.

addressBook
> This element with the `Yes` value is added to the result dataset if the HTTP request contains the `AddressBook` parameter and does not contain the `CloseBook` parameter or if the HTTP request contains the `OpenBook` parameter.

body
> This string element contains the current message body text.

mailerWidth
> This string element contains the `MailerWidth` WebUser Preferences option value.

forwardedMessage
> This optional string element contains the HTML representation of the original message. This element is added to the result dataset if the HTTP request `Operation` parameter is `Forward`.

DSN
> This element with the `Yes` value is added to the result dataset if the HTTP request contains the `DSN` parameter.

SaveSent
> This element with the `Yes` value is added to the result dataset if the WebUser Preferences contain a non-empty `SentBox` option and the HTTP does not contain the `Filled` parameter or the HTTP request contains the `SaveSent` parameter.

desiredCharset
> This string element contains the name of charset to be used in the composed message.

charset
> This element is the `UTF-8` string if the `Use UTF8` WebUser Preferences option is set to "for Reading and Composing". Otherwise, this element contains the same value as the `desiredCharset` result dataset element.

isEvent
    This element with the `Yes` value is added to the result dataset if the item being composed is a Calendar Event item.

isTask
    This element with the `Yes` value is added to the result dataset if the item being composed is a Calendar Task (ToDo) item.

isNote
    This element with the `Yes` value is added to the result dataset if the item being composed is a Note item.


The following elements are added if the item being composed is a Calendar item:

allDayEvent
    This element with the `Yes` value is added to the result dataset if the item is an All-Day Event. The element value is controlled with the HTTP parameter of the same name.

---

## Name: `MailboxSettings`

The HTTP request must contain the `Mailbox` parameter - the name of the Mailbox to manage.

**Actions**

If the HTTP request contains the `Remove` parameter, the Mailbox is removed. If the HTTP request also contains the `RemoveSub` parameter, all Mailbox submailboxes are removed, too.
If the operation has been successful and the Show Subscribed Mailboxes option is selected in the WebUser Preferences, the deleted Mailbox(es) are removed from the Account subscription list.
If the operation has been successful, the `removed` element with the `Yes` string value is added to the result data set and the code component stops request processing. Otherwise, the `errorCode` element with the operation error code string value is added to the result dataset.

If the HTTP request contains the `Rename` parameter and the `NewName` parameters is not empty, the Mailbox is renamed. The NewName parameter value is converted into the "UTF-7 Mailbox Name encoding" format and is used as the new Mailbox name.
If the HTTP request also contains the `RenameSub` parameter, all Mailbox submailboxes are renamed, too.
If the operation has been successful and the Show Subscribed Mailboxes option is selected in the WebUser Preferences, the renamed Mailbox(es) are renamed in the Account subscription list.
If the operation has been successful, the `removed` element with the `Yes` string value is added to the result data set and the code component stops request processing. Otherwise, the `errorCode` element with the operation error code string value is added to the result dataset.

If the HTTP request contains the `Update` parameter, the code component retrieves all `Acc` parameters from the request. Each `Acc` parameter should have a numeric value. For each retrieved `Acc` parameter value *nnn*, the `znnn` parameter is retrieved. If it contains a non-empty string, all *Knnn* request parameters are retrieved, where *K* is a [Mailbox access right](#) letter.
The list of *znnn* name strings with their *Knnn* parameter sets are used to form and set the new ACL list for the selected Mailbox.
If the ACL update operation has been successful, the `messageCode` element with the `Updated` string value is added to the result dataset. Otherwise, the `errorCode` element with the operation error code string value is added to the result dataset.

If the HTTP request contains the `DeleteAll` parameter, all Mailbox messages are deleted, using the method specified with the WebUser Preferences DeleteMethod option. If the operation has been successful, the `messageCode` element with the `MessagesDeleted` string value is added to the result dataset.

**Result Dataset**

renamed
    This element with the `Yes` string value is added to the dataset if the Mailbox has been renamed. In this case no other

element is added to the result dataset.

removed
> This element with the `Yes` string value is added to the dataset if the Mailbox has been removed. In this case no other element is added to the result dataset.

rights
> This array element contains the Mailbox ACL (Access Control List) elements. Each array element is a dictionary with the following elements:

> ident
>> this string element contains the ACL element *name*.

> index
>> this string element contains the element number in the ACL set.

> lookup, select, seen, flags, insert, post, create, delete, admin
>> these elements with `Yes` string values are added when the ACL element includes these Mailbox access rights.

---

### Name: `Alerts`

This code component can be called implicitly, if the Web Application module has detected a pending [Alert](#) message.

**Actions**

If the HTTP request contains the `AlertTime` parameter, that parameter should contain a time stamp in the ACAP format. The code component confirms all Alerts older than the specified time.

If the HTTP request contains the `returnURL` parameter, the parameter value is added to the result dataset (as the `returnURL` element).

**Result Dataset**

alerts
> This element is added to the result dataset if there are pending Alerts for the session user. The element value is an array of dictionary elements, each element describing one alert message. Each dictionary contains the following elements:

> time
>> A string element containing the time when the alert was posted.

> text
>> A string element containing the alert message text.

currentTime
> This element is added to the result dataset if there are pending Alerts for the session user. Its string value contains the current time in the ACAP format.

returnURL
> If the Alerts page was retrieved automatically, when some other page had to be displayed, the encoded URL for that other page is placed into this string element.

---

**Name: `Subscription`**

**Actions**

If the HTTP request contains the `Open` parameter, the `MailboxName` parameter value is converted into the "UTF-7 Mailbox Name encoding" format, the converted string is added to the result dataset as the `jump` element, and the request processing ends.

If the HTTP request contains the `Update` parameter:

- All `Elem` request parameters are retrieved, converted into the "UTF-7 Mailbox Name encoding" format, and form the new Account Subscription list.
- All `AliasName` request are retrieved, they should contain numeric values. For each retrieved numeric value *nnn*, the parameters pairs a*nnn* and m*nnn* are retrieved. If both parameters exist and contain non-empty strings, the strings are converted into the "UTF-7 Mailbox Name encoding" format, and are used to form the new set of Account Mailbox Aliases.
- If the Subscription or Mailbox Aliases update operation fails, the `errorCode` element is added to the result dataset. Otherwise the `messageCode` element with the `Updated` string value is added to the result dataset.

**Result Dataset**

`jump`
  The name of the Mailbox to open ("to jump to"). If this element exists, none of the following elements is added to the result dataset.

`subscription`
  This array element contains the Account subscription list. Each array element is a string with some Mailbox name.

`aliases`
  This array element contains the Account Mailbox Aliases list. Each array element is a dictionary with the following elements:

  `index`
    A string with this Mailbox Alias element index.

  `name`
    A string with the Mailbox Alias name.

  `ref`
    A string with the name of the Mailbox this Alias points to.

---

**Name: `Password`**

**Actions**
If the HTTP request contains the `ModifyPassword` parameter, the request should also contain the `OldPassword` parameter, and that parameter should match the current Account password. If the `OldPassword` parameter value is correct:

- The `RecoverPassword` request parameter value is set as the new `RecoverPassword` Account setting. The `messageCode` element with the `Updated` string value is added to the result dataset.
- If the Account user is allowed to modify the Account password, the `NewPassword1` and `NewPassword2` parameters are checked. If they are non-empty and match each other, then the Account password is updated using these parameters value.

If the password has been updated successfully, the `messageCode` element with the `PasswordChanged` string value is added to the result dataset. If the password update operation failed, the `errorCode` element is added to the result dataset.

## Result Dataset

RecoverPassword
> This string element contains the Account `RecoverPassword` setting value.

---

## Name: `PublicInfo`

### Actions

If the HTTP request contains the `Update` parameter, the request should also contain zero, one, or several `ID` parameters, each with a numeric value. For each ID parameter, its numeric value *nnn* is used to retrieve the pair of `Nnnn` and `Vnnn` string parameters. The value of the `Nnnn` parameter specifies the name of the Account Public Info setting, the value of the `Vnnn` parameter specifies the setting value. These pairs are used to set the new Account Public Info settings. If an empty string is specified as a setting value, the setting is removed from the Account Settings.

If the Public Info settings have been updated successfully, the `messageCode` element with the `Updated` string value is added to the result dataset. If the password update operation failed, the `errorCode` element is added to the result dataset.

### Result Dataset

publicInfo
> This array element contains a set Public Info elements. It contains one element for each Public Info Setting specified with the [Directory Integration](#) settings. Each array element is a dictionary with the following elements:
>
> id
> > this string element contains the element number in the set.
>
> name
> > this string element contains the Public Info setting name.
>
> value
> > this string element contains the current Public Info setting value. This element exists only if the Account contains this Public Info setting.

---

## Name: `WebSite`

The code component uses the generic WebSite component to process the HTTP parameters and to form the result dataset. Before the generic component is called, the following elements are added to the result dataset:

### Result Dataset

fileRef
> The `WebFile/` string.

pageRef
> The `website.wssp` string.

---

## Name: `Bye`

## Actions

The code component can delete old messages from the Trash (as specified in the WebUser preferences and if string parameter `noEmpty` was not set) and closes the session. The session will be destroyed as soon as this HTTP request is processed, so the `bye.wssp` code can use session data, but the produced HTML code should not contain references to session objects.

## Result Dataset

blockAlerts
> This element has the `Yes` string value. It is added to the result dataset to prevent Alert processing.

# Generic Code Components

The Web Application module has several generic components used to process both stateless and session requests.

### Generic `Mailbox` component

## Actions

If the HTTP request contains `Filter`, `Search`, `Limit` parameters, these parameter values are used to modify the Mailbox "viewer" current *Filter*, *Search*, *Limit* values.

If the HTTP request parameter `Skip` exists, it should have a numeric value. This number is used to set the current *first message index* - the number of the first message to be displayed on this page.

If the HTTP request contains the parameter `Next`, then the current *first message index* is increased by the current *Limit* value.

If the HTTP request contains the parameter `Prev`, then the current *first message index* is decreased by the current *Limit* value.

If the HTTP request contains the parameter `Sort`, its numeric value specifies the number of "sorting" column (to sort the Mailbox view by the first column, the `Sort` parameter should be 0).

If the HTTP request contains the parameter `SDir`, its numeric value specifies the sorting order: the value 1 requests ascending order, the value 0 - descending order, the value -1 reverses the current sorting order.

## Result Dataset

checkAll
> This element has the `CHECKED` string value. It is added to the result dataset if the HTTP request contains the `MarkAll` parameters.

filter
> The string value of this element is the current *Filter* string.

search
> The string value of this element is the current *Search* string.

limit
> The string value of this element is the current *Limit* value (a number).

sentBox
> This element has the `YES` string value and exists only if this Mailbox is a `Sent`-type Mailbox.

headers
> The array value of this element contains Mailbox view column headers. Each array element is a dictionary with the following elements:

index
> This string element contains the column number.

name
> This string element contains the column name.

hilited
> This element has the YES string value and exists only if this column is the sorting column.

sdir
> If this column is not the sorting column, this element contains the current sorting order (0 or 1). If this column is the sorting column, this element contains the reversed current sorting order (1 - *current sorting order*).

ralign
> This element has the YES string value and exists only if this is a date- or size-type column and thus needs the reversed horizontal alignment.

messages
> The array value of this element contains the Mailbox view data. Each array element is a dictionary with message data, and it contains the following elements:

id
> this element contains the message Unique ID (UID)

color
> if the message has the X-Color header field with a valid HTML "color" string as its value, this element exists and contains the value of that header field.

notText
> this optional element has the YES string value; it exists if the message Content-Type is not text.

notAltText
> this optional element has the YES string value; it exists if the message Content-Type is not text and the message Content-Type/Subtype is not multipart/alternative.

fields
> this array element contains message column data. The columns are stored in the same order as columns in the headers result dataset element. Each element is a dictionary. It contains the following elements:

hilited
> This element has the YES string value and exists only if this column is the sorting column.

sdir
> If this column is not the sorting column, this element contains the current sorting order (0 or 1) If this column is the sorting column, this element contains the reversed current sorting order (1 - *current sorting order*).

ralign
> This element has the YES string value and exists only if this is a date- or size-type column and thus needs the reversed horizontal alignment.

isRef
> This element exists for the selected column and for the first "clickable" column. If it exists, it contains the string YES.

value
> This element contains the column data. It exists for all columns except for the Status column. For the Sent and Received columns the element contains the "date" value - values of that type can be displayed using the DATE:, DATETIMESHORT and similar prefixes.

isStatus
> This YES string element exists if the column is the Status column.

isDate

This This `YES` string element element exists if the column is the Sent or Recieved column and the `value` element contains the "date"-type value.

isPty
>    This `YES` string element exists if the column is the Priority column.

status
>    This element exists if the column is the Status column. If it exists, it contains the one of the following strings:

>    - If the message has the Deleted flag - `Deleted`, otherwise
>    - If the message has the Draft flag - `Draft`, otherwise
>    - If the message has the Redirected flag - `Redirected`, otherwise
>    - If the message does not have the Seen flag - `Unread`, otherwise
>    - If the message has the Answered flag - `Answered`, otherwise
>    - `Read`

flagged
>    This element exists if the column is the Status column and the message has the Flagged flag. If it exists, it contains the string `YES`.

recent
>    This element exists if the column is the Status column and the message has the Recent flag. If it exists, it contains the string `YES`.

hidden
>    This element exists if the column is the Status column and the message has the Hidden flag. If it exists, it contains the string `YES`.

media
>    This element exists if the column is the Status column and the message has the Media flag. If it exists, it contains the string `YES`.

firstNumber
>    This string element contains the number of the first message in the view.

firstNumber1
>    This string element contains the number of the first message in the view increased by 1.

lastNumber
>    This string element contains the number of the first message in the view increased by the number of the `messages` array elements if this array is not empty, or increased by 1 if the `messages` array is empty.

numTotal
>    The total number of messages in this Mailbox.

numUnread
>    The total number of unread messages (messages without the Seen flag) in this Mailbox.

numSelected
>    The total number of Mailbox messages that can be displayed with the current Filter and Search values.

multiPage
>    This element with the `YES` string value is added if the `nSelected` value is not equal to the number of the `messages` array elements.

sortColumn
>    This element contains the number of the currently selected sorting column.

sortAscending
>    This element contains `1` is the currently selected sorting order is ascending, and `0` if it is descending.

---

**Generic `Message` component**

The generic Message component is used to convert the an RFC822 message into an HTML text. It processes simple and multi-part messages, attachments, digests, inline images and other letter components. To build a HTML presentation, the component uses [Code Components for Message Rendering](#).

---

# Code Components for Message Rendering

The Web Application module can render messages, converting them into a markup (HTML) text. This process is controlled by the Application module itself. It detects the MIME structure of the message, and processes each part recursively. For each part, a dataset is produced and a .wssp file is used to produce a markup language presentation.

Message Rendering code components do not perform any actions.

A Result Dataset produced by every Message Rendering code component includes the following fields:

MIMEPart
> this string element contains a URL reference for the message or the message part that is being rendered.

filesRef
> this string element contains the URL prefix needed to retrieve files from the proper Skin. When a message is being rendered withing some WebUser Session, this string is the same as the `SESSION(filesRef)` string.

isWML
> this string element exists and contains the `YES` string if the message should be displayed using the WML markup language.

printVersion
> this string element exists and contains the `YES` string if the message should be displayed in a printable form.


The following Message Rendering code components are implemented:

---

### Name: `RFC822Message`

This code component is used to render a mail message - a message stored in a Mailbox or a `message/rfc822` MIME subpart of some other message.

**Result Dataset**

RFC822Header
> this string element contains the rendered markup presentation of the message RFC822 header.

RFC822Body
> this string element contains the rendered markup presentation of the message RFC822/MIME body.

isSubPart
> this optional element exists and has the YES string value if the message is a MIME subpart of some other message.

---

### Name: `RFC822Header`

This code component is used to render an RFC822 mail message header.

**Result Dataset**

RFC822Fields

this element is an array with one dictionary-type element for each "visible" field in the header. Each dictionary contains the following elements:

name
> this string element contains the header field name.

value
> this string element contains the MIME-decoded field value.

---

## Name: `AttachmentPart, ImagePart`

This code component is used to render an image or an attachment. Images and attachments can be separate MIME parts, or can be embedded into text parts using UUENCODE encoding.

### Result Dataset

attachmentName
> this string contains the file name as it is stored in the message data.

fileName
> this string contains the "cleaned" file name (with all path components removed and image file name suffix added if necessary).

embeddedPart
> if this string parameter exists, the file is an "embedded" UUENCODE data, and the string specifies the embedded component number inside the MIME part.

decodedSize
> this string element contains the approximate size of the decoded file data.

---

## Name: `DeliveryReportPart`

This code component is used to render a `message/report` MIME subpart.

### Result Dataset

MessageFields
> this array element contains one dictionary element for each message-level report field. Each dictionary element contains the following elements:

name
> this string element contains the report field name.

value
> this string element contains the MIME-decoded report field value.

Reports
> this array element contains one array element for each recipient report. Each recipient report is an array containing one dictionary element for each recipient-level report field. Each dictionary element contains the following elements:

name
> this string element contains the report field name.

value
> this string element contains the MIME-decoded report field value.

## Name: `DispositionReportPart`

This code component is used to render a `message/disposition-notification` MIME subpart.

**Result Dataset**

fields
> this array element contains one dictionary element for each message-level report field. Each dictionary element contains the following elements:

> name
>> this string element contains the report field name.

> value
>> this string element contains the MIME-decoded report field value.

## Name: `EncryptedPart`

This code component is used to render an encrypted MIME subpart.

**Result Dataset**

decryptedPart
> this array element contains the rendered markup presentation of the decrypted content. The element exists only if decryption was successful.

decryptionErrorCode
> if this string element exists, it contains the error message explaining why content decryption has failed.

cipherName
> This string element contains the name of the cipher used for content encryption.

keyLength
> This string element contains the size of the encryption cipher key (in bits).

## Name: `SignedPart`

This code component is used to render a signed MIME subpart.

**Result Dataset**

signedPart
> this array element contains the rendered markup presentation of the signed content. The element exists only if decoding was successful (for binary-type signed messages).

encoding
> this string element contains words "Binary" or "Text" depending on the format of the signed subpart.

decryptionErrorCode
> if this string element exists, it contains the error message explaining why binary content decoding has failed.

digesterName
> This string element contains the name of the digester used for digital signing.

signatures
> If this array element exists, then the signed content was verified by at least one digital signature. Each element of this array is a dictionary with signature data. These dictionaries contain the following elements:

> contact

this string element contains the E-mail address of the signer

commonName
>
> this string element contains the "real name" of the signer

Country, Province, Organization, Unit
>
> these optional string elements contain additional information about the signer.

---

### Name: `CalendarPart`

This code component is used to render an iCalendar subpart.

### Result Dataset

Summary, Location, Comment
>
> these string element contain the iCalendar attribute data.

Priority
>
> this numeric string element contains the iCalendar element PRIORITY attribute value.

dateFrom
>
> this date element contains the iCalendar element DTSTART attribute value.

method
>
> this string element contains the iCalendar object METHOD parameter.

description
>
> this string element contains the formatted DESCRIPTION attribute value.

organizer
>
> this optional dictionary element contains the ORGANIZER attribute. The dictionary can contain various parameters specified for that attribute ("cn", etc.). The E-mail address (the value) of the attribute is available as the theValue element of this dictionary.

attendees
>
> this optional array element contains dictionary elements for each ATTENDEE attribute. Each dictionary can contain various parameters specified for that attribute ("cn", "role", etc.). The E-mail address (the value) of the attribute is available as the theValue element of this dictionary.

isEvent
>
> this optional element exists and contains the YES string if the iCalendar element is a VEVENT. The following optional elements may exist only if this isEvent element exists:

> allDayEvent
> >
> > this optional element exists and contains the YES string if the VEVENT is an All-Day Event.

> recurrence
> >
> > this optional element exists and contains the YES string if the VEVENT is a recurrent Event.

> duration
> >
> > this optional numeric string element exists and contains the Event duration in second if the VEVENT is a recurrent event.

> dateTill
> >
> > this optional date element exists and contains the Event "end date" if the Event is not a recurrent one, and if it's not a 1-day All-Day Event.

> busyStatus
> >
> > this optional string element contains the status of the Event if the iCalendar method is PUBLISH.

isTask
>
> this optional element exists and contains the YES string if the iCalendar element is a VTODO. The following optional elements may exist only if this isTask element exists:
>
> dateTill
>> this optional date element contains the VTODO "due date".
>
> percentComplete
>> this numeric string element contains the VTODO PERCENT-COMPLETE attribute value.

---

### Name: `vCardPart`

This code component is used to render an vCard subpart.

**Result Dataset**

FN
> this string element contains the Formatted Name vCard attribute value.

UID
> this string element contains the UID vCard attribute value.

REV
> this date element contains the REV vCard attribute value.

elements
> this array element contains dictionary elements for other vCard attributes. Each dictionary has the following elements:
>
> name
>> a string with vCard attribute name
>
> value
>> the vCard element value. The value can be a dictionary or an array of dictionary elements if vCard has several attributes of the same name. Each dictionary contains the attribute parameters and the attribute value as `theValue` element.

---

# Redirect-type Response

Session and Stateless requests processed using WSSP files produce markup language documents. Before these documents are sent to the client browser, their first lines are checked. If the first document line starts with the `<REDIRECT>` tag, the rest of the document first line is interpreted as a URL.

The Server returns the 301 ("Moved") response code with the Location header containing the specified URL.

The Server also processes the `<RELREDIRECT>` tag at the beginning of the document. It is processed in the same way as the `<REDIRECT>` tag, but the URL placed into the Location header is prefixed with the http or https prefix, the server name (and, optionally, port number) retrieved from the request URL.

---

# CG/PL Applications

Session and Stateless requests can be processed using CG/PL applications (rather than the code components built into the Server). These CG/PL applications are called Web Applications.

A stateless request addressing a */programName*`.wcgp` or */programName*`.wcgp/`*some_url* resource makes the Server load the

*programname*`.wcgp` [CG/PL](#) application from the selected Skin.
If the application produces any output, it is returned to the client browser, otherwise the 404 Not Found error is returned.

An application can be started by addressing a `/auth/`*programName*`.wcgp` or `/auth/`*programName*`.wcgp/`*some_url* resource. Such an HTTP request is authenticated first, and the application is executed on behalf of the authenticated Account.

An application can be started by addressing the `/sys/`*programName*`.wcgp` resource. The application is executed in the Server-wide or the Cluster-wide Unnamed Skin environment on behalf of the `postmaster` Account in the Main Domain. The `sysEntry` code section of the *programName*.wcgp file is executed.

Session requests addressing a `/`*programName*`.wcgp` in the Session (`/Session/`*sessionID*`/`) realm makes the Server load the *programname*`.wcgp` [CG/PL](#) application from the Session Skin.
Only the `GET`, `POST`, and `HEAD` HTTP requests are supported.
If the application produces any output, it is returned to the client browser, otherwise the 404 Not Found error is returned.
The application is executed on behalf of the Account the Session belongs to.

When a CG/PL module is loaded to process an HTTP request, the `main` module entry is executed.

Web Applications can use CG/PL [external-declarations](#). When a code section (a procedure or a function) declared as *external* is called, a file with the code section name and the `.wcgi` extension is loaded from the current Skin. The program code in this file must contain the code section with the specified name and of the proper type (a procedure or a function).
The program code in an `.wcgi` file may contain other code sections as well.

CG/PL Web Applications can use the following built-in procedures and functions.

## HTTP Request Input

`GetHTTPParameter(`*name* `[ ,`*index* `])`
>This function retrieves an HTTP request parameter. HTTP request parameters include URL parameters and/or form fields.
>The *name* value should be a string specifying the HTTP request parameter name.
>An HTTP request can contain several parameters with the same name. If the *index* parameter is present, its value should be a number specifying which HTTP request parameter of the specified name should be retrieved.
>If the HTTP request contains the `FormCharset` parameter, its value is interpreted as the character set name. The function converts the raw parameter data into the UTF-8 character set. If the `FormCharset` parameter is absent, the raw parameter data is assumed to be encoded using the UTF-8 character set.
>If the parameter value contains several lines, the EOL (line separator) symbols are converted to those used with the Server host OS.
>This function returns a string containing the parameter value. If the parameter with the specified name and, optionally, index is not found, this function returns a null-value.

`GetHTTPBinaryParameter(`*name* `[ ,`*index* `])`
>This function retrieves an HTTP request parameter raw data.
>This function returns a datablock containing the parameter value. If the parameter with the specified name and, optionally, index is not found, this function returns a null-value.

`GetHTTPField(`*name*`)`
>This function retrieves the *name* HTTP request field.
>The following fields are supported: `Authorization`, `Referer`, `Destination`, `Cookie`, `User-Agent`, `Host`.
>If the `name` value is an empty string, the HTTP request URL (without the optional URL parameters) is returned.
>If the `name` value is the `Schema` string, the request URL schema used (the `http` or `https` string) is returned.
>If the `name` value is the `Host` string, the request Host field is returned, with the optional "port" part removed.
>If the `name` value is the `Port` string, the port number from the Host field is returned; if no port is specified in that field, the number 80 is returned for clear text connections, and the number 443 for secure connections.
>If the specified field name if not in the supported set, or if the request does not contain the specified field, this function returns a null-value.

`GetHTTPQuery()`
>This function retrieves the parameter string from the HTTP request URL.
>If the request URL does not contain parameters, the function return a null-value.

`GetHTTPResource()`

This function retrieves the HTTP request query string, including all URL parameters.

`GetHTTPMethod()`

This function retrieves the HTTP request method string (`GET`, `POST`, etc.).

`GetHTTPType()`

This function retrieves the HTTP request body Content type.

`GetHTTPSubtype()`

This function retrieves the HTTP request body Content subtype.

`GetHTTPData()`

This function retrieves the HTTP request body.

If the request body content subtype is `xml` or it ends with `+xml`, the function tries to convert the body content into XML and returns an XML object.

If the request body content type is `text`, the function converts the body content into UTF-8 and returns a string.

Otherwise, the function returns a datablock.

If any conversion operation fails, or if the request body is absent or it is too long, the function return a null-value.

## HTTP Response Output

`SetHTTPResponseData(data)`

This procedure sets the HTTP response body.

The *data* value should be either a string, a datablock, a dictionary, or an XML object.

`SetHTTPResponseType(mimeType,subtype)`

This procedure sets the HTTP response Content-Type field values.

The *mimeType* and *subtype* values should be strings containing valid MIME atoms.

`SetHTTPResponseCode(code)`

This procedure sets the HTTP response code.

The *code* value should be a number in the 200..999 range or a string.

If the string starts with a 3-digit prefix and the `-` symbol, then the prefix specifies the HTTP response code, and the rest of the string specifies the HTTP response text.

If there is no such prefix in the *code* string value, then the HTTP response code is set to 500, and the entire string is used as the HTTP response text.

`RemoteIPAddress()`

This function returns an ip-address object specifying the IP Address the HTTP request was received from.

`ProxiedIPAddress()`

If the HTTP request was relayed by some external HTTP proxy, this function returns the original client IP Address.

Otherwise the function returns a null-object.

`SendHTTPContinue()`

This function sends the 100-Continue HTTP response.

If the response is successfully sent, the function returns a null-value, otherwise it returns an error code string.

`AddHTTPResponseField(fieldName,fieldValue)`

This procedure adds a field to the HTTP response.

The *fieldName* and *fieldValue* values should be strings.

`ProcessWSSP(pageName,resultSet)`

This procedure composes the HTTP response body using the specified WSSP page and the result dataset.

The *pageName* value should be a string with the page name (without the `wssp` suffix). This page is retrieved from the currently selected Skin.

The *resultSet* value should be a dictionary. The Server modifies this dictionary adding the common Session or stateless processing elements (see above), and then it is passed to the WSSP interpreter.

The resulting page is set as the HTTP response body.

## Session Data

Functions and procedures listed in this section are available only for Session HTTP requests.

`SessionData()`
      This function returns the [Session Dataset](#) dictionary.

---

# CommuniGate Pro

# Web Server-Side Programming (WSSP)

- **Scripting Elements**
- **Expressions**
- **Text Elements**
- **Structural Elements**

The CommuniGate Pro Web Application module processes a request for a WSSP file by calling a code component that produces a *dataset* - a dictionary containing text string keys and values, associated with those keys. Values can be text strings, numbers, timestamp, arrays, or dictionaries. See the Data section for more details on data types.

For example, when a Domain default page is requested, the code component is called. The component processes request (HTML FORM) parameters and produces a *dataset* - a *dictionary* containing keyed values. For example, a dataset produced with the component processing the Login requests may contain `canAutoSignup`, `hasMailLists`, and `hasCertificate` keys.

The Web Application module then uses the script code from a WSSP file to convert this dataset into a markup language (HTML, WML, etc.) page.

## Scripting Elements

The WSSP file is a markup language (usually - HTML) file with two additional types of elements:

- text elements, started and ended with double percent (`%%`) symbols
- structural elements, started with the `<!--%%` marker and ended with the `-->` marker.

The following is a sample of an WSSP document:

```
<html>
<body>

<h1>Welcome to %%server%%. Your ID is %%ID%%.</h1>

<!--%%IF EXISTS(lastLogin)-->
Last time you visited us on %%lastLogin%%
<!--%%ENDIF-->


</body>
</html>
```

This WSSP document contains the `%%server%%`, `%%ID%%`, and `%%lastLogin%%` text elements, and the `<!--%%IF EXISTS(lastLogin)-->` and `<!--%%ENDIF-->` structural elements (these text elements are fictitious, do not try to use these samples in your real .wssp pages).

If the WSSP document should contain non-ASCII symbols, the UTF-8 character set should be used. When the WSSP document is being processed, the Web Application module retrieves the `charset` string value from the produced data dictionary. If this value is not `UTF-8`, then the WSSP text is converted into this *page charset*.

# Expressions

The text and structural WSSP elements use expressions - combinations of names and symbols that specify the data to be retrieved from the data dictionary or from other available sources.

The WSSP scripting uses several types of expressions:

- data element
- array scanner
- keyed element
- indexed element
- function call
- boolean expression
- string constant

An alphanumeric string (such as `system` or `id`) is a data element name. The value of such an expression is the dataset value associated with this name. If the dataset does not have a specified key, the expression value is a null-value.
Example: the dataset contains the key `system` and its associated value is the `Sun Solaris` string, the value of the expression `system` is the string `Sun Solaris`.

The dataset dictionary is case-insensitive, so the data element names are case-insensitive, too.

An alphanumeric string followed by the `[]` symbols is interpreted as an index scanner name. It can be used only inside the `<!-- %%FOREACH name...-->` ....`<!--%%ENDFOR name-->` structure where this index element is defined (see below). The index scanner names are case-insensitive.

An expression followed by the dot (`.`) symbol and an alphanumeric string is a keyed element. The expression before the dot symbol is calculated, and its value should be a dictionary. The alphanumeric string after the dot symbol specifies the key to be used to extract the value from that dictionary. If the value of the expression before the dot symbol is not a dictionary, or if it does not contain the specified key, the keyed element value is a null-value.
Keys can be specified as quoted strings, in this case they can specify non-alphanumeric symbols.
Example: the dataset contains the key `settings` and its associated value is the 2-element dictionary: `{OS = "Sun Solaris"; CPU = "sparc";}`. The value of the `settings.OS` expression is the `Sun Solaris` string, the value of the `settings."OS1"` expression is a null-value.

An expression followed by an *index expression* in square bracket symbols (`[index]`) is an indexed element. The expression before the square bracket is calculated, and its value should be an array or a dictionary. The *index expression* is calculated, and its value should be a number or a string representing a number. This number specifies which array element or dictionary key becomes the value of this indexed expression. If the value of the *index expression* is `0`, the first array element or the first dictionary key string is retrieved .
An *index expression* can be specified as a numeric constant.
If the value of the expression before the bracket symbol is not an array or a dictionary, or if the value of the *index expression* is not a number, or if the value of the *index expression* represents a number that is negative or is equal or greater than the number of array or dictionary elements, the value of the index expression is a null-value.

An alphanumeric string followed by the `(` symbol is a function call. Elements after the `(` symbol specify the function parameters, and they are followed by thes `)` symbol.
Function names are case-insensitive.
The list below specifies the available functions and their parameters.

Boolean expressions are two expressions separated with the `|` (OR) symbol, or with the `&` (AND) symbol, or with the `^` (XOR) symbol.
You can use parentheses to enclose expressions:

```
expression1 | expression2
expression1 & expression2
(expression1 & expression2) | expression3
```

A boolean value of an expression is positive (true), if:

- the expression value is a string and it does not start with symbols `N`, `n`, `-`, or `0`, or
- the expression value is non-zero number.
- the expression value is a timestamp and it is not the "remote past" constant.

A string constant is a sequence of symbols enclosed into quote symbols. The quote symbols and the backslash symbols must be prefixed ("escaped") using the backslash `\` symbol: `"My \"test\" string"`.

SESSION(*key*)

This function can be used only in Session-based requests. The function value is the Session Dataset value associated with the string *key*. The *key* parameter can be specified as an alphanumeric string, or as a string constant.
Example: the `SESSION(accountName)` expression value is the name of the CommuniGate Pro Account this session is opened for.
The Session Dataset is case-insensitive. It contains the following keys and values:

| Key | Value |
| --- | --- |
| ID | a string with the unique identifier of this session |
| accountName | a string with the session Account name |
| domainName | a string with the name of the Domain the session Account belongs to |
| filesRef | a string with the URL prefix needed to retrieve files from the session Skin |
| fullAccountName | a string with the session Account full name: *accountName@domainName* |
| loginAddress | a string specifying the network (IP) address the user was using when initiating this session |
| loginTime | the timestamp with the session start time |
| selectableMailboxes | an array with the names of all "selectable" Mailboxes |
| addressBooks | an array with the names of all available Address Books |
| webFolders | an array with the File Storage folder names |
| selectedMailbox | a string with the name of the target Mailbox for the last Copy/Move operation |
| selectedAddressBook | a string with the name of the currently selected Address Book. |
| selectedWebFolder | a string with the name of the target File Storage folder for the last Store File In operation |
| webSiteEnabled | this `"YES"` string element exists if the storage limit for the File Storage is not set to zero |
| openMailboxes | a dictionary with all currently opened Mailboxes (each dictionary key is the Mailbox UTF8 name). |

[CG/PL Web Applications](#) can insert additional elements into the Session Dataset.

SETTINGS(*key*)

This function can be used only in Session-based requests. The function value is the effective WebUser setting value associated with the string *key*. The *key* parameter can be specified as an alphanumeric string, or as a string constant.

ACCOUNTSETTINGS(*key*)

This function can be used only in Session-based requests. The function value is the effective Account setting value associated with the string *key*. The *key* parameter can be specified as an alphanumeric string, or as a quoted string.

INCLUDEARG(*number*)

This function can be used only inside an include file. The *key* should be a decimal number specifying the parameter number of the `<!--%%INCLUDE-->` element that invoked this include file.
The first parameter has the number 0.
If the `<!--%%INCLUDE-->` element did not specify enough parameters, the function value is a null-value.

EXISTS(*expression*)

The parameter is an expression. Its value is calculated, and the function returns the string `"YES"` if the calculated value is not a null-value, or the string `"NO"` if the returned value is a null-value.

DOESNOTEXIST(*expression*)

The parameter is an expression. Its value is calculated, and the function returns the string `"NO"` if the calculated value is not a null-value, or the string `"YES"` if the returned value is a null-value.

YESNO(*expression*)

The parameter is an expression. Its boolean value is calculated, and the value is positive (true), the function returns the string `"YES"`, otherwise it returns the string `"NO"`.

`BOOLARRAY()`
> The value is a 2-element array, containing the strings `"NO"` and `"YES"`.

`NOT(`*expression*`)`
> The parameter is an expression. Its boolean value is calculated, and the function returns a null-value if the value is positive (true), otherwise the function returns the string `"YES"`.

`EQUALS(`*expression1* `AND` *expression2*`)`
> Both expressions are calculated, and if the calculated values match (including the case when both expressions return a null-value), the function returns the string `"YES"`. Otherwise the function returns a null-value.

`EQUALSNOCASE(`*expression1* `AND` *expression2*`)`
> Both expressions are calculated. The function returns the string `"YES"` if both calculated values are a null-value or if both values are strings and these strings match using the ASCII case-insensitive comparison operation. In all other cases the function returns a null-value.

`EQUALS(`*expression* `AND` *string*`)`
> The value of the expression is calculated and compared with the string, specified as a quoted string. If the value matches the string, the function returns the string `"YES"`. Otherwise the function returns a null-value.

`EQUALSNOCASE(`*expression* `AND` *string*`)`
> The value of the expression is calculated and compared with the string, specified as a quoted string. If the value matches the string using the ASCII case-insensitive comparison operation, the function returns the string `"YES"`. Otherwise the function returns a null-value.

`ISINDEX(`*expression* `IN` *scanner*`)`
> The *scanner* should be the name of the `<!--%%FOREACH` *scanner* `IN ...-->` construct surrounding the current portion of the script code. The expression value is calculated, and if its numeric value matches the current index in the array this *scanner* is used for, the function returns the string `"YES"`. Otherwise the function returns a null-value.

`ISFIRST(`*scanner*`)`
> The *scanner* should be the name of the `<!--%%FOREACH` *scanner* `IN ...-->` construct surrounding the current portion of the script code. If the current value of the array index is zero, the function returns the string `"YES"`. Otherwise the function returns a null-value.

`ISLAST(`*scanner*`)`
> The *scanner* should be the name of the `<!--%%FOREACH` *scanner* `IN ...-->` construct surrounding the current portion of the script code. If the current value of the array index is equal to the number of array or dictionary elements minus one, the function returns the string `"YES"`. Otherwise the function returns a null-value.

`ISEVEN(`*scanner*`)`
> The *scanner* should be the name of the `<!--%%FOREACH` *scanner* `IN ...-->` construct surrounding the current portion of the script code. If the current value of the index is even, the function returns the string `"YES"`. Otherwise the function returns a null-value.

`ISHALF(`*scanner*`)`
> The *scanner* should be the name of the `<!--%%FOREACH` *scanner* `IN ...-->` construct surrounding the current portion of the script code. If the current value of the index is equal to the number of array or dictionary elements divided by 2, the function returns the string `"YES"`. Otherwise the function returns a null-value.

`CHECKED(`*expression*`)`
> The parameter is an expression. Its boolean value is calculated, and if its positive (true), the function returns the string `"checked"`, otherwise the function returns a null-value.

`HASPARENTMAILBOX(`*expression*`)`
> The parameter is an expression. Its value is calculated, and the function returns the string `"YES"` if the calculated value is a string, representing a name of hierarchical Mailbox. Otherwise the function returns a null-value.

`ISSTRING(`*expression*`)`
> The parameter is an expression. Its value is calculated, and the function returns the string `"YES"` if the calculated value is a string. Otherwise the function returns a null-value.

`ISNUMBER(`*`expression`*`)`
> The parameter is an expression. Its value is calculated, and the function returns the string `"YES"` if the calculated value is a number. Otherwise the function returns a null-value.

`ISARRAY(`*`expression`*`)`
> The parameter is an expression. Its value is calculated, and the function returns the string `"YES"` if the calculated value is an array. Otherwise the function returns a null-value.

`ISDICTIONARY(`*`expression`*`)`
> The parameter is an expression. Its value is calculated, and the function returns the string `"YES"` if the calculated value is a dictionary. Otherwise the function returns a null-value.

`ISDATE(`*`expression`*`)`
> The parameter is an expression. Its value is calculated, and the function returns the string `"YES"` if the calculated value is a timestamp. Otherwise the function returns a null-value.

`ISDATA(`*`expression`*`)`
> The parameter is an expression. Its value is calculated, and the function returns the string `"YES"` if the calculated value is a datablock object. Otherwise the function returns a null-value.

`NULL()`
> The value of this function is a null-value.

`EMPTYSTRING()`
> The value of this function is an empty string.

`EMPTYARRAY()`
> The value of this function is an array with zero elements.

`EMPTYDICTIONARY()`
> The value of this function is an empty dictionary.

`CURRENTTIME()`
> This function value is a timestamp - the current global time.

`SELECTEDLANGUAGE()`
> This function value is a string - the currently selected language name.

`SELECTEDTIMEZONE()`
> This function value is a string - the currently selected time zone name.

`STRING(`*`key`*`)`
> The value of this function is the object associated with the *key* in the Skin Text Dataset. This object should be a string, otherwise the function returns a null-value. The key can be specified either as a quoted string literal, or as an expression - the expression value is calculated and used as the key.

`DICTIONARY(`*`key`*`)`
> The value of this function is the object associated with the *key* in the Skin Text Dataset. This object should be a dictionary, otherwise the function returns a null-value. The key can be specified either as a quoted string literal, or as an expression - the expression value is calculated and used as the key.

`ARRAY(`*`key`*`)`
> The value of this function is the object associated with the *key* in the Skin Text Dataset. This object should be an array, otherwise the function returns a null-value. The key can be specified either as a quoted string literal, or as an expression - the expression value is calculated and used as the key.

`TRANSLATE(`*`string`* `USING` *`dictionary`*`)`
> The *string* parameter is an expression that should return a string value; the *dictionary* parameter is an expression that should return a dictionary value. If the dictionary contains a string value for the key specified with the first parameter value, the function returns this string. Otherwise the value of the *string* parameter is returned;
> Example: the dataset contains the element `boxName` with the string value `INBOX`, and the element `boxNames` with the dictionary value `{INBOX = Incoming; Trash = "Trash Can";}`. The value of the `TRANSLATE(boxName USING boxNames)` expression is the `"Incoming"` string.

`CONTAINS(`*`string`* `IN` *`array`*`)`
> The function returns the string `"YES"` if the value of the *array* parameter is an array, and the string is equal to one of the

array elements. Otherwise the function returns a null-value.

The string can be specified either as a quoted string literal, or as an expression.

RANDOMELEMENT(*array*)

The *array* parameter is an expression that should return an array value; The value of this function is a randomly-selected element from that array.

MONTHNAMES()

The function returns a fixed array with 12 string elements:
`("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec")`.

WEEKDAYS() and WEEKDAYS(*expression*)

If expression is not specified, the function returns a fixed array with 7 string elements: (`Sun,Mon,Tue,Wed,Thu,Fri,Sat`).

If expression is specified, its result should be a string with one of the weekday names, and the function returns an array with 7 weekday names, starting with the specified weekday:

> `WEEKDAYS(startOfWeek)`
> returns `("Tue","Wed","Thu","Fri","Sat","Sun","Mon")` if the value of the startOfWeek variable is `"Tue"`.

KNOWNCHARSETS()

The function returns a fixed array with string elements - names of character sets known to the system (`"ISO-8059-1","ISO-2022-jp","KOI8-R", ...)`.

KNOWNTIMEZONES()

The function returns a fixed array with string elements - names of time zones known to the system
`("NorthAmerica/Pacific","Europe/Central","(+0900) Japan/Korea", ...)`

REQUESTSECURE()

The function returns the string `"YES"` if the current HTTP request is secured (i.e. is sent via the HTTPS protocol), otherwise the function returns a null-value.

REQUESTRESOURCE()

The function returns the resource string (the local part of the URL with optional URL parameters) of the current HTTP request.

REQUESTSOURCEIP()

The function returns the string with the IP Network Address the HTTP request originated from.

SERVERVERSION()

The function returns the string with the current CommuniGate Pro Server version.

---

# Text Elements

Text elements are specified using double percent markers. The body of a text element is an expression with an optional prefix.

%%*expression*%%

The expression is calculated. If the expression value is a string or a number, it substitutes substitutes the text element in the resulting markup code.

Otherwise the entire text element is removed from the resulting markup code.

%%HTML:*expression*%%

The expression is calculated. If the expression value is a string or a number, it substitutes substitutes the text element using HTML escape symbols. A string value is converted from the UTF-8 charset into the required charset.

Otherwise the entire text element is removed from the resulting markup code.

Example:

> if the expression result is the `">=GO=>"` string, the text element is substituted with:

```
&gt;=GO=&gt;
```

`%%HTMLUTF8:`*expression*`%%`

    The expression is calculated. If the expression value is a string or a number, it substitutes substitutes the text element using HTML escape symbols (the string is not converted from the UTF-8 charset into the page charset).
Otherwise the entire text element is removed from the resulting markup code.

    Example:

        if the expression result is the `>=GO=>` string, the text element is substituted with:

```
&gt;=GO=&gt;
```

`%%URL:`*expression*`%%`

    The expression is calculated. If the expression value is a string or a number, it substitutes the text element using URL escape symbols.
Otherwise the entire text element is removed from the resulting markup code.

    Example:

        if the expression result is the `"Stop It?"` string, the text element is substituted with:

```
Stop%20It%3F
```

`%%MAILBOXRAWNAME:`*expression*`%%`

    The expression is calculated. If the value is not a string, then the entire text element is removed from the resulting markup code. If the result is a string, it is converted from the IMAP-specified Mailbox name encoding into the UTF-8 charset, then it is converted into the required charset, and the converted string substitutes the text element using HTML escape symbols.

`%%MAILBOXNAME:`*expression*`%%`

    The expression is calculated. If the value is not a string, then the entire text element is removed from the resulting markup code. If the result is a string X, the `TRANSLATE(X USING DICTIONARY("MailboxNames"))` expression is calculated. Then the prefix works in the same way as the `MAILBOXRAWNAME:` prefix.

`%%MAILBOXLASTNAME:`*expression*`%%`

    The expression is calculated. If the value is not a string, then the entire text element is removed from the resulting markup code. If the result is a string X, it is interpreted as a Mailbox name. If the Mailbox name is a hierarchical one, only the last part of the name is used, otherwise the entire name is used. The name (or its last part) is converted in the same way as for the `MAILBOXNAME:` prefix.

`%%URLMAILBOXPARENT:`*expression*`%%`

    The expression is calculated. If the value is not a string, then the entire text element is removed from the resulting markup code. If the result is a string X, it is interpreted as a Mailbox name. If the Mailbox name is a hierarchical one, the name of the parent Mailbox is used. It is converted in the same way as for the `URL:` prefix. If the Mailbox name is not a hierarchical one, the element is removed from the resulting markup code.

`%%JAVASCRIPT:`*expression*`%%`

    The expression is calculated. If the value is not a string or a number, then the entire text element is removed from the resulting markup code. If the result is a string or a number, it substitutes the text element using escape symbols needed for JavaScript strings. The result is converted into Unicode and all non-ASCII symbols are presented using the `\u`*hhhh* escape sequences.
Note: this prefix does not put any quotation marks around the string.

    Example:

        if the expression result is the `'What do "they" think?'` string, the text element is substituted with:

```
What do \"they\" think
```

`%%JSON:`*`expression`*`%%`

      The expression is calculated. The expression is converted to a string using JSON (JavaScript Object Notation) representation, and this string substitutes the text element. Timestamp objects are represented as RFC822-formatted date strings.

      Example:

            if the expression result is the `( 'What do "they" think?', #123)` array, the text element is substituted with:

```
["What do \"they\" think",124]
```

`%%SIZE:`*`expression`*`%%`

      The expression is calculated. If the value is not a string or a number, then the entire text element is removed from the resulting markup code.

      If the expression value is a string, it is converted into a numeric value (the number of bytes).

      The string should contain some number and, optionally, the `k` or `K` suffix (in this case the number is multiplied by 1024), or the `m` or `M` suffix (in this case the number is multiplied by 1048576).

      Alternatively, a string can start with a letter `u` or `U`, in this case the converted number of bytes is -1.

      If the expression value is a number, its numeric value is used.

      The resulting number is converted into a "size string", using the dictionary retrieved with the `DICTIONARY("SizePictures")` expression. If the number is negative, the dictionary is used to translate the string `unlimited`, and the result is used to replace this text element using the same conversions as used for the a text element with the `HTML:` prefix.

      The calculated number of bytes is checked to see if it represents an even number of Megabyte or Kilobytes, and that number is greater than one. Then a string value associated with the keys `"M"`, `"K"`, or `""` (empty string) is retrieved from the dictionary. The string is expected to contain the `^0` symbol combination which is replaced with the number of megabytes, Kilobytes, or bytes specified with the *expression* value. The resulting string is processed with the method used for the `HTML:` text element prefix.

      If the `DICTIONARY("SizePictures")` expression result is a null-value, or this dictionary does not contain a string value for the required key, the resulting string is built using the number and the key name (`20M`, `1345K`, `182345777`).

`%%ROUNDSIZE:`*`expression`*`%%`

      This prefix works in the same way as the `SIZE:` prefix, but the numeric *expression* value can be modified: if the value is equal or larger than 10000, then it is converted into "Kilo" (value = value / 1024 * 1024), and if the value is equal or larger than 10240000, it is converted to "Mega" (value = value / 1048576 * 1048576).

`%%TIME:`*`expression`*`%%`

      The expression is calculated.

      If the result is a null-value, then the entire text element is removed from the resulting markup code.

      If the result is a number, its value is interpreted as the number of seconds.

      If the result is a string, it is converted into a numeric value (the number of seconds):

      The string should contain some number and, optionally, the `s` or `S` suffix, or the `m` or `M` suffix (in this case the number is multiplied by 60), or the `h` or `H` suffix (in this case the number is multiplied by 3600), or the `d` or `D` suffix (in this case the number is multiplied by 86400).

      The resulting number is converted into a "time string", using the dictionary retrieved with the `DICTIONARY("TimePictures")` expression.

      The calculated number of seconds is checked to see if it represents an even number of weeks, days, hours, or minutes, and if that number is greater than 1. Then a string value associated with the keys `weeks`, `days`, `hours`, `minutes`, or `seconds` is retrieved from the dictionary. The string is expected to contain the `^0` symbol combination which is replaced with the number of weeks, days, hours, minutes, or seconds specified with the *expression* value. The resulting string is converted from the UTF-8 into the required charset and the converted string substitutes the text element using HTML escape symbols.

      If the `DICTIONARY("TimePictures")` expression result is a null-value, or this dictionary does not contain a string value for the required key, the resulting string is built using the number, a space, and the key name (`3 weeks`, `11 hours`, `5 seconds`).

      Example:

            If the data element `elapsedTime` is the `2400` string, then the text element

```
%%TIME:elapsedTime%%
```

      will be substituted with the following string:

```
40 minutes
```

If the `DICTIONARY("TimePictures")` exists and contains the string `"^0mins"` as the `minutes` value, then the text element

```
%%TIME:elapsedTime%%
```

will be substituted with the

```
40mins
```

string.


`%%DATETIME(`*`formatName`*`):`*`expression`*`%%`
> The expression is calculated. If the value is not of the timestamp type, then the entire text element is removed from the resulting markup code. If the value is a timestamp, it is converted into a text string using the specified format.
> The *format* parameter can be specified as an alphanumeric atom, or as a quoted string.
> The format string is the result of the `DICTIONARY("DatePictures").`*`formatName`* expression. If this expression does not result in a string value, the `((^h:^m:^s ^W ^D-^M-^Y))` string is used instead.
> The format string is processed by replacing the following symbol combinations with the actual timestamp value parts:

| symbols | substituted with |
|---|---|
| `^D` | the day of month (2-digit) |
| `^d` | the day of month (1- or 2-digit) |
| `^M` | the month name (one of those returned with the MONTHNAMES() function), translated using `DICTIONARY("DatePictures")` |
| `^N` | the month number (2-digit, from `01` to `12`) |
| `^Y` | the year number (2-digit) |
| `^y` | the year number (4-digit) |
| `^s` | the seconds value (2-digit) |
| `^m` | the minutes value (2-digit) |
| `^H` | the hours value (2-digit), from `00` to `23` |
| `^h` | the hours value (1- or 2-digit), from `12,1` to `11` |
| `^t` | the `AM` or `PM` string, translated using `DICTIONARY("DatePictures")` |
| `^w` | the weekday number (Sun - 0) |
| `^W` | the weekday name (one of those returned with the WEEKDAYS() function), translated using `DICTIONARY("DatePictures")` |

> The resulting string is placed into the markup code using the `HTML:` prefix processing.

`%%LOCALDATETIME(`*`formatName`*`):`*`expression`*`%%`
> Processing is the same as for the `DATETIME(`*`formatName`*`)` prefix, but the timestamp value is converted into the local time first.

`%%DATETIMEAS(`*`format`*`):`*`expression`*`%%`
> The expression is calculated. If the value is not of the timestamp type, then the entire text element is removed from the resulting markup code. If the value is a timestamp, it is converted into a text string using the specified format.
> The *format* parameter should be a quoted string or an expression with a string value. This string is used as the format string.

`%%LOCALDATETIMEAS(`*`format`*`):`*`expression`*`%%`
> Processing is the same as for the `DATETIMEAS(`*`format`*`)` prefix, but the timestamp value is converted into the local time first.

`%%DATE:`*`expression`*`%%`
> Processing is the same as for the `DATETIME("dateOnly")` prefix.

`%%LOCALDATE:`*`expression`*`%%`

Processing is the same as for the `LOCALDATETIME("dateOnly")` prefix.

`%%DATEWEEK:`*`expression`*`%%`

Processing is the same as for the `DATETIME("dayAndDate")` prefix.

`%%DATETIME:`*`expression`*`%%`

Processing is the same as for the `DATETIME("dateAndTime")` prefix.

`%%LOCALDATETIME:`*`expression`*`%%`

Processing is the same as for the `LOCALDATETIME("dateAndTime")` prefix.

`%%DATETIMESHORT:`*`expression`*`%%`

Processing is the same as for:

- the `DATETIME("timeOnly")` prefix if the timestamp value specified with the expression is "very close" to the current time (the absolute difference is less than 22 hours),
- the `DATETIME("monthDate")` prefix if the timestamp value specified with the expression is "close" to the current date (the absolute difference is less than 180 days),
- the `DATETIME("dateOnly")` prefix in all other cases.

`%%LOCALDATETIMESHORT:`*`expression`*`%%`

Processing is the same as for the `DATETIMESHORT:` prefix, but the the expression value is converted into the local time first.

`%%HTMLTRUNCATED(`*`number`*`):`*`expression`*`%%`

The expression is calculated. If the value is not a string, then the entire text element is removed from the resulting markup code. If the result is a string, the string is truncated, then it is converted from the UTF-8 charset into the required charset, and the converted string substitutes the text element using HTML escape symbols. Not more than the *number* of symbols ("glyphs") are substituted. If the string has more glyphs, the ...symbols are added after the string.
Example: if the expression result is the `"Test Subject"` string, the text element substituted with the HTMLTRUNCATED(10) prefix is

```
Test Subje..
```

`%%HTMLTRUNCATED(`*`expression1`*`):`*`expression`*`%%`

The same as the above, but the *expression1* parameter is an expression that is calculated. The *expression1* should have a numeric value. This value specifies the number of glyphs to be taken from the *expression* string.
Example: `HTMLTRUNCATED(STRING("FieldLength")):theField`

`%%HTMLSUBST(`*`parameter0`*`,`*`parameter1`*`,...):`*`expression`*`%%`

The all parameterN expressions and the expression is calculated. If the value of the expression is not a string, then the entire text element is removed from the resulting markup code. If the result is a string, all its ^N substrings are replaced with the string values of parameterN expression (`^0` substrings are replaced with the value of parameter0, `^1` substrings are replaced with the value of parameter1, etc.). If the parameter value is not a string, the ^N substring is removed.
The resulting string is converted from the UTF-8 charset into the required charset, and the converted string substitutes the text element using HTML escape symbols.
Example: if the `text1` element of the Text Dataset is the `"My String1"` string, the var2 element of the Result Dataset is `My Var2`, and the `text2` element of the Text Dataset is the `"comparing ^0 & ^1"` string, then the following code:

```
%%HTMLSUBST(STRING("text1"),var2):STRING("text2")%%
```

will be substituted with

```
comparing My String1 &amp; My Var2
```

.

`%%URLSUBST(`*`parameter0`*`,`*`parameter1`*`,...):`*`expression`*`%%`

The same as the above, but the resulting string substitutes this text element using URL escape symbols.

`%%INDEX:`*`scanner`*`%%`

The *scanner* should be the name of the `<!--%%FOREACH` *`scanner`* `IN ...-->` construct surrounding the current portion of the script code. The scanner index decimal value substitutes this text element. For the first element the value 0 is used.

`%%INDEX1:`*`scanner`*`%%`
> The *scanner* should be the name of the `<!--%%FOREACH` *`scanner`* `IN ...-->` construct surrounding the current portion of the script code. The scanner index decimal value substitutes this text element. For the first element the value 1 is used.

`%%DUMP:`*`expression`*`%%`
> The expression is calculated. And the value [textual representation](#) substitutes this text element.

`%%LENGTH:`*`expression`*`%%`
> The expression is calculated.
> If the value is a string, the string length (in bytes) substitutes this text element.
> If the value is an array, the number of array elements substitutes this text element.
> If the value is a dictionary, the number of dictionary key-value pairs substitutes this text element.
> Otherwise, this text element is removed.

`%%CENTS:`*`expression`*`%%`
> The expression is calculated. Its numeric value is converted into a numeric string with at least 3 decimal digits, and the period (`.`) symbol is inserted before the last 2 decimal digits. The resulting string substitutes this text element.

---

# Structural Elements

The structural elements start with the `<!--%%` symbols and end with the `-->` symbols. The structural elements themselves are always removed from the resulting markup code.

`<!--%%IF` *`expression`*`-->`
> This structural element may be followed with one or more
> `<!--%%ELIF` *`expression`*`-->`
> element(s), then it may be followed with the
> `<!--%%ELSE-->`
> element, and then it must be followed with the
> `<!--%%ENDIF-->`
> element.
>
> The boolean value of the expression in the IF element is calculated, then the values of the expressions in the ELIF elements (if any exists) are calculated, till one of those values is positive (true).
> If an element with such a value is found, the portion of the script between this element and the following ELIF, ELSE, or ENDIF element is processed.
> If an element is not found, and there is no ELSE element, the script between the IF and ENDIF elements is removed completely, otherwise the portion between the ELSE and ENDIF elements is processed.
>
> Example:

```
<!--%%IF EXISTS(lastLogin)-->We have not seen you since <i>%%HTML:lastLogin%%<iI>
<!--%%ELSE-->Welcome, new user!
<!--%%ENDIF-->
```

> In this example, if the dataset contains the `lastLogin` element with the `20-Apr-2007` string value, this script portion will produce
> ```
> We have not seen you since <i>20-Apr-2007</i>
> ```
>
> If the dataset does not contain the `lastLogin` element, this script portion will produce
> ```
> Welcome, new user!
> ```

`<!--%%FOREACH` *`scanner`* `in` *`expression`*`-->`
or
`<!--%%FORALL` *`scanner`* `in` *`expression`*`-->`
> This structural element can be followed with the
> `<!--%%EMPTYFOR` *`scanner`*`-->`
> element, and must be followed with the

```
<!--%%ENDFOR scanner-->
```
element. All elements should have the same *scanner* alphanumeric string.

The value of the *expression* is calculated. The resulting value should be an array or a dictionary.
The portion of the script between the `<!--%%FOREACH scanner ...-->` and `<!--%%EMPTYFOR scanner-->` elements or (if the
`<!--%%EMTPYFOR scanner-->` element does not exist) the portion of the script between the `<!--%%FOREACH scanner ...-->`
and `<!--%%ENDFOR scanner-->` elements is processed repeatedly, for each array or dictionary element.

Expressions specified in that portion of the script (called the "loop body") can use the `scanner[]` *scanner reference* to
access the current element of the *expression* value.
If the *expression* is a dictionary, then the `scanner[*]` *key reference* can be used to access the current element key.
If the *expression* value is not an array or dictionary, or if it is an empty array or dictionary, and the `<!--%%EMPTYFOR`
`scanner-->` element is specified, the portion of the script between the `<!--%%EMPTYFOR scanner-->` and `<!--%%ENDFOR`
`scanner-->` elements is processed.

Example:

```
<table border="1">
<tr><td>File Name</td><td>File Size</td></tr>
<!--%%FOREACH elem in fileList-->
<tr><td>%%HTML:elem[].name%%</td><td>%%elem[].size%%</td></tr>
<!--%%EMPTYFOR elem-->
<tr><td colspan="0"> </td></tr>
<!--%%ENDFOR elem-->
</table>
```

In this example, the data element `fileList` is expected to be an array of dictionaries. Each dictionary is expected to
contain string values for keys `name` and `size`.
If the `fileList` value is

`({name=MyReport; size=2300;},{name="My Old Report"; size=4000;})`

then this portion of the script will produce the following HTML code:

```
<table border="1">
<tr><td>File Name</td><td>File Size</td></tr>
<tr><td>MyReport</td><td>2300</td></tr>
<tr><td>My Old Report</td><td>4000</td></tr>
</table>
```

The `<!--%%IF ...-->` ...`<!--%%ELSE-->` ...`<!--%%ENDIF-->` constructs and the `<!--%%FOREACH ...-->` ... `<!--%%ENDFOR ...--`
`>` constructs can be nested.

`<!--%%FOREACHINC scanner in expression-->`
The same as the `<!--%%FOREACH` construct, but it repeats the "loop body" code portion one time more than the number of
elements in the *expression* value array or dictionary.
For this additional run, the `scanner[*]` and `scanner[]` values are null-values.
If the *expression* value is not an array or a dictionary, the loop body is not repeated, and the portion between the `<!--`
`%%EMPTYFOR scanner-->` and `<!--%%ENDFOR scanner-->` elements (if any) is processed.

`<!--%%FOREACHREV scanner in expression-->`
The same as the `<!--%%FOREACH` construct, but it repeats the "loop body" code portion for each element in the *expression*
value array or dictionary, taken in the reversed order (the last element first).

`<!--%%INCLUDE filename[( parameter1 [, parameter2 ... ])]-->`
*filename* should be a quoted string or an expression with a string value.
The file with the *filename* name is retrieved from the same Skin this script is retrieved from. The file should contain some
WSSP code. This code is executed within the current context (using the same dataset).
The resulting markup code is used to replace this structural element.

It is recommended to use the `.wssi` file name extension for files designed to be used with the INCLUDE element.

The INCLUDE elements can be nested, this means that `.wssi` files can include other `.wssi` files.

Unlike the `#include` operators in the C and C++ languages, this operator is a real operator, not a pre-processor operator. As a result, if an `<!--%%INCLUDE filename-->` element is used within a `<!--%%FOREACH ...-->` ... `<!--%%ENDFOR ...-->` construct, the *filename* code can be executed several times, once for each element of the array used in the FOREACH construct.

You can specify one or more parameters, enclosing them into parentheses and separating them with the comma symbol. Each parameter is an expression that is evaluated before the *filename* code is executed.
The *filename* code can reference its parameter values using the `INCLUDEARG` function.

`<!--%%NUMERICMENU` *selected* [DEFAULT *selectedDefault* ] IN (*number1*,*number2*,....,*numberN*) [ DISPLAY *dictionary*]`-->`
This element is substituted with a sequence of the `<option value="`*value*`">`*presentation* string elements.
>The number of elements and the *value* used for each element are defined by the list of numeric numbers - *number1*,*number2*,....,*numberN*. These numbers should be specified in the ascending order, and these numbers should not be less than -1.
The *selected* expression is calculated, and its value should be a string. The string numeric value is used to add the keyword `selected` to the `<option value="`*value*`">` element that has the same *value*.
The `DISPLAY` keyword and the *dictionary* expression can be omitted. In this case, the *presentation* strings are the same as the *value* strings, this means that these strings are numbers.

Example:
The dataset element `sizeLimit` is the `200` string.
The element:

```
<!--%%NUMERICMENU sizeLimit IN (-1,0,100,200,300)-->
```

will be substituted with the following markup text:

```
<option value="-1">-1<option value="0">0
<option value="100">100<option value="200" selected>200<option value="300">300
```

If the `DISPLAY` keyword and the *dictionary* expression are specified, the expression is calculated. If the expression value is a dictionary, then the *presentation* strings are the numeric values "translated" using this dictionary, the results are converted into the required charset and placed using the HTML escape symbols.

Example:
The dataset element `sizeLimit` is the `200` string.
The Skin Text Dataset contains the `Limits` dictionary: `{"-1" = Unlimited; 0 = "Off & Shut";}`.
The element:

```
<!--%%NUMERICMENU sizeLimit IN (-1,0,100,200,300) DISPLAY DICTIONARY("Limits")-->
```

will be substituted with the following markup text:

```
<option value="-1">Unlimited<option value="0">Off &amp; Shut
<option value="100">100<option value="200" selected>200<option value="300">300
```

If the keyword `DEFAULT` with the *selectedDefault* expression are specified, an additional `<option value="-2">`*defaultPresentation* string is added before the sequence. If the *selected* expression value is a null-value, this string element will have the keyword `selected` added.

The *defaultPresentation* string is the `DefaultValuePicture` string retrieved from the Skin Text Dataset. This string should

contain the `^0` symbol combination. This symbol combination is substituted with the *selectedDefault* expression value. If the DISPLAY dictionary is specified, the *selectedDefault* expression value is translated first.

Example:

> The dataset element `sizeLimit` is the `200` string.
> The dataset element `defLimit` is the `-1` string.
> The Skin Text Dataset contains the `DefaultValuePicture` string: `default(^0)`.
> The Skin Text Dataset contains the `Limits` dictionary: `{"-1" = Unlimited; 0 = "Off & Shut";}`.
> The element:

```
<!--%%NUMERICMENU sizeLimit DEFAULT defLimit IN (-1,0,100,200,300) DISPLAY DICTIONARY("Limits")-
-->
```

> will be substituted with the following markup (HTML) text:

```
<option value="-2">default(Unlimited)
<option value="-1">Unlimited<option value="0">Off &amp; Shut
<option value="100">100<option value="200" selected>200<option value="300">300
```

`<!--%%TIMEMENU` *selected* [`DEFAULT` *selectedDefault* ] `IN` (*time1,time2,....,timeN*) [ `DISPLAY` *dictionary*]`-->`

> This element is processed in the same way as the NUMERICMENU element. The *timeN* units specify time intervals converted to numeric values (the number of seconds) using the same algorithm as the algorithm used for the `TIME:` text elements.
> The *selected* and *selectedDefault* expressions should return strings. Each string is converted into a numeric value (the number of seconds) using the same algorithm as the algorithm used for the `TIME:` text elements.
> Values are translated using the specified `DISPLAY` dictionary. If the `DISPLAY` dictionary is not specified or it does not contain a string for the given value, the *presentation* time strings are composed using the same method as the method used for the `TIME:` text elements.

`<!--%%SIZEMENU` *selected* [`DEFAULT` *selectedDefault* ] `IN` (*size1,size2,....,sizeN*) [ `DISPLAY` *dictionary*]`-->`

> This element is processed in the same way as the NUMERICMENU element. The *sizeN* units specify data sizes converted to numeric values (the number of bytes) using the same algorithm as the algorithm used for the `SIZE:` text elements.
> The *selected* and *selectedDefault* expressions should return strings. Each string is converted into a numeric value (the number of bytes) using the same algorithm as the algorithm used for the `SIZE:` text elements.
> Values are translated using the specified `DISPLAY` dictionary. If the `DISPLAY` dictionary is not specified or it does not contain a string for the given value, the *presentation* size strings are composed using the same method as the method used for the `SIZE:` text elements.

`<!--%%ENUMMENU` *selected* [`DEFAULT` *selectedDefault* ] `IN` *valueSet* [ `DISPLAY` *dictionary*]`-->`

> This element is substituted with a sequence of the `<option value="`*value*`">`*presentation* string elements.

> The number of elements and the *value* used for each element are defined by the value of the *valueSet* expression. This value should be an array of strings. The *value* in each element is the string index in the valueSet array result.

> The *selected* expression is calculated, and its value should be a string. The keyword `selected` is added to the `<option value="`*value*`">` element created for the the *valueSet* array element that matches the *selected* expression result.

> The `DISPLAY` keyword and the *dictionary* expression can be omitted. In this case, the *presentation* strings are the same as the *valueSet* result elements.

Example:

> The dataset element `color` is the `"Green"` string.
> The dataset element `colors` is the `("Blue", "Green", "Red")` array.
> The element:

```
<!--%%ENUMMENU color IN colors-->
```

> will be substituted with the following markup (HTML) text:

```
<option value="0">Blue<option value="1" selected>Green<option value="2">Red
```

If the DISPLAY keyword and the *dictionary* expression are specified, the expression is calculated. If the expression value is a dictionary, it is used to "translate" the *valueSet* array strings before converting them into the required charset and placing into the resulting markup text using the HTML escape symbols.

Example:

The dataset element color is the "Green" string.
The dataset element colors is the ("Blue", "Green", "Red") array.
The Skin Text Dataset contains the Colors dictionary: {Blue = "Night Blue"; Green = "Grass Green";}.
The element:

```
<!--%%ENUMMENU color IN colors DISPLAY DICTIONARY("Colors")-->
```

will be substituted with the following markup (HTML) text:

```
<option value="0">Night Blue<option value="1" selected>Grass Green<option value="2">Red
```

If the keyword DEFAULT with the *selectedDefault* expression are specified, an additional <option value="-1">*defaultPresentation* string is added before the sequence. If the *selected* expression value is a null-value, this string element will have the keyword selected added.

The *defaultPresentation* string is the DefaultValuePicture string retrieved from the Skin Text Dataset. This string should contain the ^0 symbol combination. This symbol combination is substituted with the *selectedDefault* expression value. If the DISPLAY dictionary is specified, the *selectedDefault* expression value is translated first.

Example:
The dataset element color is the "Green" string.
The dataset element defColor is the "Blue" string.
The dataset element colors is the ("Blue", "Green", "Red") array.
The Skin Text Dataset contains the DefaultValuePicture string: default(^0).
The Skin Text Dataset contains the Colors dictionary: {Blue = "Night Blue"; Green = "Grass Green";}.
The element:

```
<!--%%ENUMMENU color DEFAULT defColor IN colors DISPLAY DICTIONARY("Colors")-->
```

will be substituted with the following HTML text:

```
<option value="-1">default(Night Blue)<option value="0">Night Blue
<option value="1" selected>Grass Green<option value="2">Red
```

```
<!--%%BOOLMENU selected [DEFAULT selectedDefault ] [ DISPLAY dictionary]-->
```
This element is substituted with a sequence of the <option value="*value*">*presentation* string elements in the same way the ENUMMENU element is processed.
Unlike the ENUMMENU element, this element does not contain the IN *valueSet* part: the built-in array ("NO","YES") array is used instead.

```
<!--%%MAILBOXMENU selected [DEFAULT selectedDefault ] IN mailboxList [NOINBOX ]-->
```
This element is substituted with a sequence of the <option value="*value*">*presentation* string elements in the same way the ENUMMENU element is processed.

The values of the *selected* and *selectedDefault* expressions are converted in the same way as they are converted for a text element with the `MAILBOXNAME:` prefix, using the `MailboxNames` dictionary from the Skin Text Dataset, this is why MAILBOXMENU elements do not have the DISPLAY part.
If the NOINBOX keyword is specified, the INBOX Mailbox (if exists in *mailboxList*) is not displayed.

`<!--%%DAYTIMEMENU selected [DEFAULT selectedDefault ][PERIOD timePeriod ] -->`
This element is substituted with a sequence of the `<option value="value">presentation` string elements to form a time-of-day menu. Each value is the time in seconds, and presentation is a string presenting this time using the "hourMinute" format (or the "^H:^m" format string, if the "hourMinute" element is not found in the `DICTIONARY("DatePictures")` data. The time values are generated from the midnight (`00:00`) till 23:59 with the timePeriod step (specified in minutes).
If the PERIOD keyword and the timePeriod value are not specified, the 30 minutes (1800 seconds) step is used.
If the PERIOD keyword is specified, the timePeriod value can be either a numeric constant specifying the period in minutes, or an expression. The expression value is calculated and converted to a number. This number specifies the period in seconds.
The *selected* and *selectedDefault* expressions should have numeric values - the currently selected (and the default) time-of-day (seconds from midnight).

`<!--%%CALENDARTIMEMENU selected [PERIOD timePeriod ] -->`
This element creates the same HTML menu as the DAYTIMEMENU element.
The *selected* expression value should be a timestamp. The time part of that value is used as the currently selected time-of-day value.

`<!--%%LOCALCALENDARTIMEMENU selected [PERIOD timePeriod ] -->`
This element creates the same HTML menu as the DAYTIMEMENU element.
The *selected* expression value should be a timestamp value. The value is converted to the local time zone, and time part of the converted value is used as the currently selected time-of-day value.

`<!--%%STRINGMENU selected [DEFAULT selectedDefault ] IN valueSet [ DISPLAY dictionary]-->`
This element works in the same way as the ENUMMENU element, but the values used are the actual data values, not their numbers as in the ENUMMENU element. The default value has the string value `default`.

`<!--%%CALENDARDATECONTROL selected NAME name [DAYSBEFORE before][DAYSAFTER after][CANNEVER ] -->`
This element composes a control or set of controls for a calendar date specified using the *selected* expression. The expression value should be a timestamp, otherwise the whole element is removed from the resulting markup code.

If the specified time is more than *before* days earlier than the current date or if it is more than *after* days later than the current date, then this element is substituted with a text with several text controls. Otherwise a menu control is composed.

The *before* and *after* values should be specified as numbers.
If the *before* value is not explicitly specified, it is set to 7.
If the *after* value is not explicitly specified, it is set to 31.
If the CANNEVER keyword is used, the control menu displays the `Never` item with the *remote future* value. If the selected value is equal to the *remote future* value, a menu control is composed.

The *name* element should be a string.

To compose text controls, the `dateOnly` format string is taken from the `DatePictures` dictionary, and the text control codes are used to substitute its ^D, ^M, ^Y, and ^y symbol combinations. Each control has the specified *name* with a prefix: the day control has the `name-D` name, the month control has the `name-M` name, the 2-digit year control has the `name-Y` name, and the 4-digit year control has the `name-y` name.

If a menu is to be composed the `select` markup language element is generated. It has the specified *name* name. The menu contains an element for the current date, *before* elements for the previous dates, and *after* elements for the future dates. The elements are formatted using the `dayAndMonthDate` format string from the `DatePictures` dictionary.
An additional element with the `...` text is added for the "after+1" date. When this element is selected, the selected date value moves outside the "menu-covered" range, allowing the user to use text controls and select an arbitrary date.

`<!--%%LOCALCALENDARDATECONTROL selected NAME name [DAYSBEFORE before][DAYSAFTER after] -->`
This element creates the same markup code as the CALENDARDATECONTROL element.

The *selected* expression value should be a timestamp. The value is assumed to be expressed in the global (GMT) terms, so the value is converted to the local time first.

Applications  Data  CLI/API  Rules  CG/PL  PBXApp  XIMSS  WebApp  WSSP  Helpers

# Helper Applications

- **Helper Settings**
- **Helper Protocol**
- **External Authentication**
- **External Message Filters**
- **External RADIUS Helpers**
- **External CDR Processor**
- **External Load Balancer**
- **External Application Helpers**
- **External Banner System**

The CommuniGate Pro Server can use external programs to implement various operations - message scanning, user authentication, RADIUS login policies, etc. All these external programs are handled in the same way, and they communicate with the Server using a simple Helper Interface protocol.

See the System Administration section to learn how to specify the Helper programs to use.

## Helper Settings

To specify the External Helper program path and other parameters, open the General page in the Settings realm of the WebAdmin Interface and click the Helpers link:

| *Helper Name* | |
|---|---|
| Log Level:  Problems | Program Path: |
| Time-out:  15 sec | Auto-Restart:  10 sec |

The checkbox next to the Helper name tells the Server to start the specified program as a separate OS process.

Log
> Use this setting to specify the type of information the Helper support module should put in the Server Log. Each Helper uses its own tag for its Log record.

Program Path
> The file name (path) of the Helper program. If a relative path is specified, it should be relative to the CommuniGate Pro *base directory*.
> If the first setting symbol is $, then the rest of the setting string should be a path relative to the CommuniGate Pro *application directory*.

Time-out
> If the Helper program does not send a response within the specified period of time, the program is stopped.

Auto-Restart
> If the Helper program stops, and this option is disabled, all pending requests are rejected. If the Helper program stops when

this option is enabled, the Server waits for the specified period of time, restarts the Helper Program and re-sends the requests to it.

# Helper Protocol

When the Helper program is launched, the Server sends commands to the Helper process via the process *standard input*. The Server reads the program responses from the process *standard output*.

Commands and responses are text lines, ending with the EOL symbol(s) used in the Server OS.

Each command starts with a sequence number, and the response produced with the Helper program starts with the same number. This method allows the Helper program to process several requests simultaneously, and it can return responses in any order.

The Helper program can send information responses at any time. An information response starts with the asterisk (*) symbol. The Server ignores information responses, but they can be seen in the Server Log.

The response lines generated with a Helper program should not be larger than 4096 bytes.

**Note:** communication between the Server and an Helper program takes place via OS *pipes*, and many programming libraries buffer output data sent to pipes. Check that your Helper program uses some form of the *flush* command after it sends a response to its standard output, otherwise the response will not reach the Server.

Helper programs are started with the CommuniGate Pro *base directory* as their current directory.

Helper programs should not write anything to their *standard error* streams, unless they want to report a reason for the failure before quitting. CommuniGate Pro reads the program *standard error* stream only after the program has terminated, and if the program writes into that stream while processing Server commands, the program will be suspended by the OS when the *standard error* pipe buffer is full.

The Interface Version command is used to provide compatibility between different versions of Helper programs and different versions of the CommuniGate Pro Server. The Server sends this command specifying the protocol version it implements:
*nnnnnn* INTF *serverInterfaceVersion*
where:

> *nnnnnn*
> > a unique sequence number for this request
> *serverInterfaceVersion*
> > the version of the Helper protocol implemented by this version of the CommuniGate Pro Server

The Helper program should return the INTF response and the supported protocol version.
*nnnnnn* INTF *programInterfaceVersion*
If the returned number is smaller than the Server protocol version, the Server will use this (older) protocol version.

When the Server shuts down or when it needs to stop the Helper program, it sends the QUIT command, and then closes the process *standard input*. The Helper program should send the OK response and it should quit within 5 seconds.

Sample session (I: - server commands sent to the program standard input, O: - responses the program writes to its standard output, COMMAND - a Helper-specific command):

```
O: * My Helper program started
I: 00001 INTF 1
O: 00001 INTF 1
I: 00002 COMMAND parameters
O: 00002 OK
I: 00003 COMMAND parameters
I: 00004 COMMAND parameters
O: * processing 00003 will take some time
O: 00004 ERROR description
O: 00003 OK
I: 00005 QUIT
O: * processed: 5 requests. Quitting.
O: 00005 OK
```

```
       I: stdin closed
```

The sample above shows that the Server does not wait for a response before it sends the next command, and that it can accept responses for several pending commands in any order - as long as each command receives a response within the specified time limit.

---

# External Authentication

External Authenticator programs can be used to provide authentication, provisioning, and routing services using external data sources.

The External Authenticator Interface protocol is based on the generic Helper Protocol.

This manual describes the External Authenticator Interface Version 11.

When a user should be authenticated using the *clear text* method, the Server sends the following command:
*nnnnnn* VRFY (*mode*) *name@domain password* [*loginAddress*]
where:

> *nnnnnn*
>> a unique sequence number for this request
>
> *mode*
>> the name of the service (IMAP, POP, FTP, etc.) that requested this authentication operation.
>> This parameter can be absent if the request has been received from an unnamed Server component.
>> If the service name is specified, it is enclosed into the parenthesis.
>
> *name*
>> user Account name
>
> *domain*
>> user Account Domain name
>
> *password*
>> password string to verify. If the password contains special symbols, the password is encoded as a quoted String.
>
> *loginAddress*
>> the network address from which the user logs in.
>> This parameter can be absent if the password is checked by an internal Server component.
>> If the parameter is specified, it is enclosed into square brackets.

When a user should be authenticated using a secure SASL method, the following command is sent:
*nnnnnn* SASL(*method*) (*mode*) *name@domain password key* [*loginAddress*]
where:

> *method*
>> the name of the secure SASL method used (CRAM-MD5, APOP)
>
> *key*
>> the *challenge* string sent to the client mailer. If the challenge contains special symbols, it is encoded as a quoted String.

If the password is accepted, the External Authenticator should return a positive response:
*nnnnnn* OK

If the password was not accepted, a negative response should be returned:
*nnnnnn* ERROR *optional-error-message*

If the password is accepted, and there is an authentication response to be returned to the client, a positive response with a quoted string should be returned:
*nnnnnn* RETURN "*authentication-response*"

SASL password verification requires that the External Authenticator program correctly implements all supported SASL methods and algorithms. Alternatively, the External Authenticator program can return the user plain text password, making the Server verify the password and calculate necessary authentication responses. The user plain text password should be returned as a quoted string:

```
nnnnnn PLAIN "plain-text-password"
```

Sample session (I: - server commands sent to the program standard input, O: - responses the program writes to its standard output):

```
I: 00001 INTF 1
O: 00001 INTF 1
I: 00010 VRFY user1@domain1.com dsyui134
O: 00010 OK
I: 00011 VRFY (IMAP) user2@domain2.com jskj23#45 [10.0.3.4]
O: 00011 ERROR incorrect password
I: 00012 SASL(CRAM-MD6) user4@domain2.com hdkj547812329394055 <pop-23456@mydomain.com> [10.0.1.4]
I: 00013 VRFY (IMAP) user2@domain2.com "jskj23\"45"
O: 00013 OK
O: 00012 ERROR unsupported SASL method
I: 00014 SASL(DIGEST-MD5) user4@domain2.com 012345 "user:qop:zz:mmm:uri" [10.0.1.4]
O: 00014 RETURN "0123456789AAAA"
I: 00015 SASL(DIGEST-MD5) user4@domain2.com 012345 "user:qop:zz:mmm:uri" [10.0.1.4]
O: 00015 PLAIN "my$$password"
```

The External Authentication program can be used to retrieve plain text passwords from external databases.

To retrieve a password, the Server sends the following command:
```
nnnnnn READPLAIN name@domain
```
where:

> *name@domain*
>> the full name (address) of the target Account.

The program should return the user plain text password as a quoted string:
```
nnnnnn PLAIN "plain-text-password"
```
If the program cannot retrieve the plain text password, it should return the FAILURE response.

The External Authentication program can be used to process unknown names, too. For example, the program can consult an external database, check if the user exists in that database, create an Account, Alias, Group, Mailing List, or Forwarder using the CommuniGate Pro CLI/API, and return a positive response to the Server. In this case, CommuniGate Pro will re-try to open a Domain object with the specified name.

To check an unknown name, the Server sends the following command:
```
nnnnnn NEW name@domain relayType
```
where:

> *relayType*
>> [MAIL] if the command is sent as a part of mail-type routing process,
>> [SIGNAL] if the command is sent as a part of signal-type routing process,
>> [ACCESS] if the command is sent as a part of access-type routing process.
> *name@domain*
>> the full name (address) of the unknown local object.

If the program sends the OK response, the Server tries to find the *name* object in the *domain* Domain again.

If the program sends the ROUTED *address* response, the Server takes the supplied *address* response and restarts the Router procedure with this address. The routed address gets a "can Relay" attribute, unless it is specified with the [NORELAY] prefix.

If the program sends the FAILURE response, the Server Router returns a "temporary internal error" code (this code causes the SMTP module to return a 4xx error code, not a permanent 5xx error code).

If the program sends any other response, the Server Router returns the "unknown user account" error.

Sample session:

```
I: 00010 NEW user1@domain1.com [MAIL]
O: 00010 ERROR this account is not known
I: 00011 NEW user2@domain2.com [MAIL]
I: 00012 NEW user3@domain2.com [ACCESS]
O: 00012 OK
O: 00011 ROUTED [NORELAY] userX@domain2.com
```

The Consult External for Unknown [Domain Setting](#) tells the Server to use the External Authenticator program when an unknown name is addressed.

The External Authentication program can be used to assist in address [Routing](#). If an address is routed to the `@external` domain, the address "local part" is passed to the External Authentication program using the ROUTE command:
```
nnnnnn ROUTE <address> relayType
```
where:

> *relayType*
>> `[MAIL]` if the command is sent as a part of mail-type routing process,
>> `[SIGNAL]` if the command is sent as a part of signal-type routing process,
>> `[ACCESS]` if the command is sent as a part of access-type routing process.
> *address*
>> the local part of the address with the `external` domain part.

If the program sends the `ROUTED` *address* response, the Server takes the supplied *address* response and restarts the Router procedure with this address. The routed address gets a "can Relay" attribute if the address is specified with the `[RELAY]` prefix.

If the program sends the `FAILURE` response, the Server Router returns a "temporary internal error" code (this code causes the SMTP module to return a 4xx error code, not a permanent 5xx error code).

If the program sends any other response, the Server Router returns the "cannot route the address" error.

Sample session:

```
I: 00010 ROUTE <user1> [MAIL]
O: 00010 ERROR this account is blocked
I: 00011 ROUTE <user2%domain1.dom> [MAIL]
I: 00012 ROUTE <"user3##name"%domain2.dom> [SIGNAL]
O: 00012 FAILURE internal error
O: 00011 ROUTED [RELAY] userX@domain100.dom
```

The External Authentication program can be used to assist in provisioning operations. If the [Consult External on Provision](#) Domain Setting is enabled, the Server sends the following commands to the External Authentication program:

before an Account is created:
> `nnnnnn PRECREATE [`*authAccount*`] `*accountName@domainName accountType initialSettings*
> *initialSettings* is a dictionary.
> If this operation fails, the Account is not created.

after an Account is created:
> `nnnnnn POSTCREATE [`*authAccount*`] `*accountName@domainName accountType initialSettings*
> If this operation fails, the newly created Account is removed.

before an Account is renamed:
> `nnnnnn PRERENAME [`*authAccount*`] `*accountName@domainName newAccountName@newDomainName*
> If this operation fails, the Account is not renamed.

after an Account is renamed:
> `nnnnnn POSTRENAME [`*authAccount*`] `*accountName@domainName newAccountName@newDomainName*
> If this operation fails, the renamed Account is renamed back.

before an Account is removed:
> `nnnnnn PREDELETE [`*authAccount*`] `*accountName@domainName*
> If this operation fails, the Account is not removed.

after an Account is removed:
> `nnnnnn POSTDELETE [`*authAccount*`] `*accountName@domainName*

before an Account License Class is changed:
> `nnnnnn PRETYPECHANGE [`*authAccount*`] `*accountName@domainName newClass*
> If this operation fails, the Account License Class is not changed.

after an Account License Class is changed:

> *nnnnnn* POSTTYPECHANGE [*authAccount*] *accountName@domainName newClass*

before an Account settings are updated:

> *nnnnnn* PREUPDATE [*authAccount*] *accountName@domainName newSettings*
>
> If this operation fails, the Account settings are not updated.

after an Account settings are updated:

> *nnnnnn* POSTUPDATE [*authAccount*] *accountName@domainName newSettings*

before an Account password is updated:

> *nnnnnn* PREPWDCHANGE [*authAccount*] *accountName@domainName newPassword*
>
> If this operation fails, the Account password is not updated.

The password is one of Account settings, so the PREUPDATE command will follow this one.

*authAccount* is the name of the Account performing the provisioning operation. If it is not known, this account name and the square brackets surrounding it are omitted.

The program should either send an OK response, or a FAILURE "*errorCode*" response.

---

# External Message Filters

External Message Filter programs are used for content filtering (anti-virus and anti-spam filtering).

The External Filter Interface protocol is based on the generic Helper Protocol.
This section describes the External Filter protocol *version 4*.

- The program should process the External Filter requests:

```
seqNum FILE fileName
```

  where *fileName* is the name of the file the program should scan.

- If message processing should proceed, the response line should have the following format:

```
seqNum [ modifiers ] OK
```

  where *modifiers* is zero, one, or several keywords (each with an optional parameter), separated using the space symbol:

  ADDHEADER *header-field-text*
  > *header-field-text* is a text string to be added to the message header. It should contain one or several RFC822/RFC2822 header fields.
  > This (optionally multi-line) header field text should be placed into the response line using the String format.

  MIRRORTO *address*
  > *address* is a text string to mirror the message to.
  > The string should contain exactly one E-mail address. To mirror the message to several addresses, use several MIRRORTO modifiers.
  > The address text should be placed into the response line using the String format.

  ADDROUTE *address*
  > This modifier is supported when the Filter is used in Server-wide and Cluster-wide Rules only.
  > *address* is a text string to add to the message "route set", i.e. to the list of addresses this message is to be delivered to.
  > The string should contain exactly one E-mail address. To add several addresses, use several ADDROUTE modifiers.
  > The address text should be placed into the response line using the String format.

  Examples:

```
1077 OK
1078 ADDHEADER "X-SPAM-SCORE: 100\eX-SPAM-FILTER: CGateProSpamFilter(r)" OK
1079 ADDHEADER "X-SPAM-SCORE: 100" MIRRORTO "spamreport@mydomain.com" MIRRORTO "abuse@mydomain.com" OK
```

- If a message should be rejected the response line should have the following format:

```
seqNum ERROR report
```

> where *report* is a text string explaining why the message is rejected.
> This (optionally multi-line) text should be placed into the response line in the CommuniGate Pro [String](#) format.

- If a message should be discarded the response line should have the following format:

```
seqNum DISCARD
```

- If message processing should be postponed (because of the license limitations, for example), the response line should have the following format:

```
seqNum REJECTED report
```

> where *report* is a text string explaining why the message processing should be postponed.

- If the program receives a request it cannot process, it should return the FAILURE response:

```
seqNum FAILURE
```

  If the program has sent the FAILURE response or any unlisted response, the Server places a record into its Log and proceeds as if it has received the OK response.
- The program SHOULD be ready to process several requests simultaneously (using several threads). Since the External Filter program is used with the Server-Wide Rules (processed with the [Enqueuer](#) Server component), the program should be ready to handle N concurrent requests, when N is the number of Enqueuer "processors" (threads).
- The program MAY be implemented as a single-threaded one, so it reads the next request only after the previous request has been processed. But this design can result in severe performance degradation of the entire Server: when a single-threaded External Filter program is scanning a large message, other messages are not being enqueued.

If the external program crashes, CommuniGate Pro suspends the Enqueuer process until the external program is restarted.

---

# External RADIUS Helpers

[External RADIUS](#) programs can be used to provide additional checks for authentication operations performed via the [RADIUS module](#), as well as to add additional attributes into RADIUS responses.

The External RADIUS Interface protocol is based on the generic [Helper Protocol](#).
This manual describes the External RADIUS Interface Version 2.

If the External RADIUS program is enabled, it is used after the user password is verified. The Server sends it the following command:
*nnnnnn* LOGIN *name@domain attributes settings*
where:

> *nnnnnn*
> > a unique sequence number for this request
> *name*
> > user Account name
> *domain*
> > user Account Domain name
> *attributes*
> > a [dictionary](#) with all request attributes.
> *settings*
> > a [dictionary](#) with the Account settings.

If the login request is accepted, the Helper program should return a positive response:
*nnnnnn* ACCEPT *attributes*

where:

> *nnnnnn*
>> the request sequence number
>
> *attributes*
>> a [dictionary](#) with the attributes to be added to the RADIUS response.

If the password was not accepted, a negative response should be returned:
`nnnnnn REJECT optional-error-message`

If the External RADIUS program is enabled, it is used to process the `Start`, `Stop`, and `Interim-Update` accounting requests. The Server sends the following command:
`nnnnnn ACCNT command name@domain attributes`
where:

> *nnnnnn*
>> a unique sequence number for this request
>
> *command*
>> the accounting command (`started`, `ended`, `updated`)
>
> *name*
>> user Account name
>
> *domain*
>> user Account Domain name
>
> *attributes*
>> a [dictionary](#) with request attributes.

The Helper program should return a positive response:
`nnnnnn OK`
where:

> *nnnnnn*
>> the request sequence number

The attributes in dictionaries should use the attribute type numeric values as keys (for example `27` for `Session-Timeout`).

The following attributes are interpreted as 32-bit integer values and they are encoded as numeric strings in dictionaries:
`NAS-Port`, `Service-Type`, `Framed-Protocol`, `Framed-Routing`, `Framed-MTU`, `Framed-Compression`, `Login-Service`, `Login-TCP-Port`, `Framed-IPX-Network`, `Session-Timeout`, `Idle-Timeout`, `Termination-Action`, `Framed-AppleTalk-Link`, `Framed-AppleTalk-Network`, `Event-Timestamp`, `NAS-Port-Type`, `Port-Limit`, `ARAP-Zone-Access`, `Password-Retry`, `Prompt`, `Tunnel-Type`, `Tunnel-Medium-Type`, `Tunnel-Preference`, `Acct-Interim-Interval`, `Acct-Delay-Time`, `Acct-Input-Octets`, `Acct-Output-Octets`, `Acct-Authentic`, `Acct-Session-Time`, `Acct-Input-Packets`, `Acct-Output-Packets`, `Acct-Terminate-Cause`, `Acct-Link-Count`, `Acct-Input-Gigawords`, `Acct-Output-Gigawords`.

The following attributes are interpreted as 32-bit IP addresses and they are encoded as *aaa.bbb.ccc.ddd* strings in dictionaries:
`NAS-IP-Address`, `Framed-IP-Address`, `Framed-IP-Netmask`, `Login-IP-Host`.

The following attributes are ignored in Helper responses:
`User-Name`, `User-Password`, `CHAP-Password`, `State`, `Proxy-State`, `EAP-Message`, `Message-Authenticator`, `Acct-Status-Type`.

All other attribute values are encoded either as a [String](#) or as [DataBlocks](#). The Server uses the DataBlocks format for those attribute values that contain bytes outside the hexadecimal 0x20-0x7F range.
The DataBlock format must be used if the value contains binary zero bytes.

If an attribute has multiple values, the attribute value is encoded as an [Array](#).

The following attributes are added to the attribute dictionaries passed to the Helper:

`0`
> the RADIUS protocol request ID. It can be used to detect retransmitted packets (duplicate requests)

`secretKey` (authentication requests only)
> a string value with the RADIUS module "shared secret" setting.

`authData` (authentication requests only)

a 16-byte DataBlock containing the "authentication data" portion of the RADIUS request.

The Vendor-specific attributes are presented using keys with negative numeric values. The key absolute value is the "VendorID" value. For each VendorID, the associated element is a dictionary. This dictionary keys are Vendor-specific "vendor types", with associated "specific attributes". The "specific attributes" values can be stored as String, DataBlocks, Number, or Arrays of Strings, DataBlocks, and/or Numbers.

Sample session (I: - server commands sent to the program standard input, O: - responses the program writes to its standard output):

```
I: 00001 INTF 1
O: 00001 OK 1
I: 00002 LOGIN user1@domain1.com {0=#15; 1="User1";4=10.0.0.1;32="NAS 1";31=4153837164;"-
311"={9=#777;10="ZZZ";}; authData=[AbndghAbndgh1sjkjkss3T=]; secretKey=a123;} {RealName="User";
NATIP="192.168.1.3";}
O: 00002 ACCEPT {8=192.168.1.3; 9=255.255.255.0; 13=(0,3);}
I: 00003 LOGIN user1@domain1.com {0=#16; 1="uSEr1";32="NAS 2";31=415.5512.12; 8=192.168.1.3;
authData=[Abnd278sjkljsljkjksFG=]; secretKey=a123;} {NATIP="10.0.1.114";}
O: 00003 REJECT
I: 00004 ACCNT started user1@domain1.com {0=#17;1="uSEr1";32="NAS 2";31=415.5512.12; 8=192.168.1.3;}
O: 00004 OK
```

Note: the Server can send several concurrent requests for the same target Account.

Note: the External RADIUS program is called when the target Account is open. In a Dynamic Cluster system this means that External RADIUS programs should run on backend servers, and that External RADIUS programs running on different backend servers will never get requests for the same Account at the same time.

# External CDR Processor

External CDR Processor programs can be used to process CDRs (Call Detail Records) generated with the Signal component when a call is attempted, established, or tiered down. They can also process CDR records generated with CG/PL applications.

The External CDR Processor Interface protocol is based on the generic Helper Protocol.
This manual describes the External CDR Processor Interface Version 1.

If the External CDR Processor program is enabled, the Signal module generated CDRs and sends them to the program.

When a CDR is composed, the Server sends the following command:
*nnnnnn* `CDR` *cdr_data*
where:

*nnnnnn*

a unique sequence number for this request
*cdr_data*

CDR data in the Signal component format or in the CG/PL Application format.

when the record is processed, the program should return a positive response:
*nnnnnn* `OK`

# External Load Balancer

External Load Balancer programs are used to control software load balancers for the Dynamic Cluster installations. These programs should be installed and enabled on all Cluster members that can act as load balancers. For each "balancing groups" the Cluster Controller selects one of the available load balancers and activates it, while other load balancers work in the "backup" mode.

The External Load Balancer Interface protocol is based on the generic Helper Protocol.
This manual describes the External Load Balancer Interface Version 1.

When the Cluster Controller detects a change in the cluster member belonging to this Load Balancer Group, the program receives the following command:

```
nnnnnn MEMBERS (ip-address [,ip-address...] ) (ip-address [,ip-address...] )
```

where:

> *nnnnnn*
>> a unique sequence number for this request
>
> *ip-addresses*
>> Two sets of Network IP addresses. The first one lists all currently available cluster members, the second list - all currently disabled (but running) cluster members. Each set may contain zero or more addresses, separated with the comma (,) symbol. Each address is the "Server WAN IPv4 Address" of the cluster member, as specified in its Network Settings.

When the program starts, or when the Server detects a change in its "Server WAN IPv4 Address" Network Setting, the program receives the following command:

```
nnnnnn LOCAL ip-address
```

where:

> *nnnnnn*
>> a unique sequence number for this request
>
> *ip-address*
>> the current "Server WAN IPv4 Address" Network Setting of this Server.

When this load balancer needs to be activated, the program receives the following command:

```
nnnnnn STARTBALANCER
```

When the Cluster Controller needs to deactivate this load balancer, the program receives the following command:

```
nnnnnn STOPBALANCER
```

---

# External Application Helpers

External Application Helper programs can be used to provide CG/PL applications with arbitrary data..

The External Application Helper Interface protocol is based on the generic Helper Protocol.
This manual describes the External Application Helper Interface Version 1.

When a client requests data, the Server sends the following command:

```
nnnnnn REQ requestData
```

where:

> *nnnnnn*
>> a unique sequence number for this request
>
> *requestData*
>> the request data from the application, which may be a string, array, dictionary or datablock.

When the request is processed, the program should return a positive response:

```
nnnnnn RESP responseData
```

where:

> *responseData*
>> the text presentation of the response data.

Sample session (I: - server commands sent to the program standard input, O: - responses the program writes to its standard output):

```
I: 00001 INTF 1
O: 00001 INTF 1
I: 00010 REQ "What time is it?"
O: 00010 RESP "11:18pm"
```

If the External Application Helper program is not running, any request immediately produces an empty response.

# External Banner System

External Banner System programs can be used to provide XIMSS and other clients with "banner" data (advertising information that a client presents to the user).

The External Banner System Interface protocol is based on the generic Helper Protocol.
This manual describes the External Banner System Interface Version 1.

When a client requests a banner, the Server sends the following command:
*nnnnnn* BANNER *bannerType* [ *accountName@domainName* ] [ INFO *bannerSetting* ] [ PREFS *bannerPreference* ] [
PARAM *paramData* ]
where:

> *nnnnnn*
>> a unique sequence number for this request
> *bannerType*
>> if the banner type string specified with the client application (specifies the client application and its banner type, for example: `prontoEmailTop`, `myClientLeftBanner`).
> *accountName@domainName*
>> full name of the Account requesting a banner.
> *bannerSetting*
>> (optional) the `BannerInfo` Account Setting value.
> *bannerPreference*
>> (optional) the `BannerClass` Account Preference value.
> *paramData*
>> (optional) the text presentation of the parameter object specified with the client application.

When the record is processed, the program should return a positive response:
*nnnnnn* RESULT *resultData*
where:

> *resultData*
>> the text presentation of the banner info.

The program can also return a blocking response:
*nnnnnn* BLOCK
The specified *bannerType* is added to the list of "blocked" types. If any client requests a banner of a "blocked" type, an empty response is returned immediately, without calling the External Banner System program.

If the External Banner System program is not running, any banner request immediately produces an empty response.

| Introduction |
| Installation |
| SysAdmin |
| Network |
| Objects |
| Storage |
| E-mail |
| Real-Time |
| Access |
| Services |
| Directory |
| Clusters |
| Applications |
| Miscellaneous |
| Licensing |
| |
| WebMail |
| Pronto! |
| PBX |

| E-Mail | Billing |

# Miscellaneous: E-Mail

- **Return Receipts**
- **Address Testing**
- **Adding Required Headers**
- **Legacy Mail Emulation**

This section describes various CommuniGate Pro E-mail Transfer features not mentioned elsewhere.

# Return Receipts

Senders can request return-receipts by including the `Return-Receipt-To:` header fields into messages. When a message containing a Return-Receipt-To: header field is delivered to a local Account, the Server generates a Delivery Notification message. That message is sent to the Return-Path address of the message, not to the address specified in the message Return-Receipt-To: header field.

# Address Testing

If a message has the
`X-Special-Delivery: test`
header field, the SMTP and Local Delivery modules do not send the message to its recipients.

The SMTP module connects to all hosts the message is addressed to, then the module sends all recipient addresses to those hosts, but it does not send the message itself.

The Local Delivery module checks if the account exists, but the module does not try to apply the Account Rules to the message and the module does not store the message in the Account Inbox.

This feature can be used to verify addresses on large mailing lists: if an address contains an unknown domain name, or the host is not unreachable, or if the host rejects a user address, an error message is generated in the regular way, and can be used to detect "bad" addresses and to "clean" the mailing list.

# Adding Required Headers

If a message does not have a properly composed RFC header part, the Server adds an RFC header to the message. This header contains the required fields only.

If a submitted message does not have a `Date:` header field, the Server adds one using the date and time when the message was submitted to the Server.

If a submitted message does not have a `Message-ID:` header field, and the message was received from a "trusted source", the Server adds a `Message-ID:` field to the message.

# Legacy Mail Emulation

The CommuniGate Pro software package includes the command-line program `mail` (`mail.exe` for the Microsoft Windows platforms). You can use this program to submit messages to the CommuniGate Pro system, as you used the legacy `mail` program to submit messages to the `sendmail` MTA.

```
mail [-EiInv]  [-d base-directory]
     [-s subject] [-f from-address]
     [-c Cc-addresses] [-b bcc-addresses] to-addresses
```

`-i, -I, -n, -v`
> These options are ignored; they are included for compatibility only.

`-E`
> Do not send messages with an empty body. This is useful for piping errors from cron scripts.

`-f from-address`
> Use the *from-address* as the message From: address. If this parameter is not specified, the current user name is used.

`-d base-directory`
> Use the *base-directory* path as the location of the CommuniGate Pro *base directory*.

`-s subject`
> Specifies the *subject* (only the first argument after the -s flag is used as a subject; be careful to quote subjects containing spaces).

`-c cc-addresses`
> Send carbon copies to the *cc-addresses*; cc-addresses should be a comma-separated list of e-mail addresses.

`-b bcc-addresses`
> Send blind carbon copies to the *bcc-addresses*; bcc-addresses should be a comma-separated list of e-mail addresses.

`to-addresses`
> A comma-separated list of E-mail addresses.

The CommuniGate Pro software package includes the command-line program **sendmail** (`sendmail.exe` for the Microsoft Windows platforms). You can use this program to submit messages to the CommuniGate Pro system, using

the interface of the legacy `sendmail` program.

```
sendmail [-i] [-t] [-d base-directory]
         [-f from-address] [-F sender-name] [-V envid]
         [-Oparameter] [-oparameter] [-B body_type] [address, ...]
```

`-d base-directory`
>   Use the *base-directory* path as the location of the CommuniGate Pro *base directory*.

`-i`

>   Ignore dots alone on lines by themselves in incoming messages. This should be set if you are reading data from a file.

`-t`

>   Read message for recipients. To:, Cc:, and Bcc: lines will be scanned for recipient addresses. The Bcc: lines will be deleted before transmission. The addresses listed on the command line will be excluded from the list of the recipients.

`-ffrom-address`
>   Use the *from-address* as the message From: address. If this parameter is not specified, the current user name is used.

`-Fsender-name`
>   Set the full name of the sender.

`-V envid`
>   The Envelope ID of the message.


`-Oparameter`
`-oparameter`
`-B body_type`
`-C config_file`
`-N dsn`
`-h hop_count`
`-R return`
`-qparameter`
>   These options are ignored.

`addresses`
>   The destination addresses (without the `-t` option) or the addresses to be excluded from the the destination address list (if the `-t` option is specified).


The CommuniGate Pro mail and sendmail commands use the [Submitted folder](#) feature.

To submit messages from your OS/400 (IBM iSeries) programs, check the [CommuniGate Pro Sendmail API for OS/400](#) document.

---

| Introduction |
| Installation |
| SysAdmin |
| Network |
| Objects |
| Storage |
| E-mail |
| Real-Time |
| Access |
| Services |
| Directory |
| Clusters |
| Applications |
| **Miscellaneous** |
| Licensing |
| |
| WebMail |
| Pronto! |
| PBX |

| E-Mail | Billing |
|--------|---------|

# Billing

- **Reservations**
- **Interfaces**
- **Operations**

The CommuniGate Pro Server can act as a Billing platform.

Each Account can have one or several *Balances*. CommuniGate Pro modules and components can post charges and credits to these balances, make refunds, etc.

The *current amount* is maintained for each Balance, along with active *reservations*.

The log of all transactions is kept in the Account data files. The Log format provides for easy backup and restore operations, and for Balance recovery after hardware failures.

The "default" or "generic" Balance uses an empty string as its name.

Monetary values (funds) are presented as 64-bit signed integer values.

# Reservations

The Balance reservations are used when some billed activity takes time and/or a billing activity takes place in several steps.

The application first "reserves" a certain monetary amount, sufficient to pay for some initial stage of the billed activity. For example, it can be an amount needed to pay for the first minute of a call, or to play the first level of a game.
If the selected Balance does not have sufficient funds, the reservation request is rejected.
Otherwise, a reservation (with an application-supplied or server-generated name) is created inside the Balance. The total funds available in the Balance are decreased by the reserved amount, but not charge record is created in the transaction log.

As the payment amount of the billed activity approaches the reserved amount, the application "extends" the created reservation by some additional amount. If the Balance does have enough "free funds" (i.e. funds not reserved with other reservations), the reservation amount is extended.

The application can release the created reservation. The reservation is removed from the Balance, and the reserved amount is returned to the Balance "free funds". For example, this function can be used when a phone call was not connected, as the reserved funds must be released without recording any charge transaction.

Finally, the application charges the Balance, specifying the reservation. The charge amount should not exceed the amount reserved.

The transaction log record is created, and the charged amount is subtracted from the Balance and from the reservation reserved amount. Optionally, the reservation can be released, and the remaining funds returned to the Balance "free funds".

When a reservation is created or extended, the application can specify the reservation expiration time. The reservation is released automatically after the specified time. This feature can be used if the application can disconnect or quit without explicitly releasing its reservations.

When creating or extending a reservation, the application can specify the amount that should be charged if the reservation is timed-out and it is released automatically. For example, when a game application extends the reservation to cover the next game level, it can specify the amount that should be already be paid for the current game. If the application quits or disconnects without explicitly charging the Balance reservation, the reservation will be released automatically, but the specified amount will be changed first.

# Interfaces

The CommuniGate Pro Billing subsystem is available for:

- CG/PL server-side applications (PBX Tasks, WebUser applets, etc.)
- XIMSS client applications.
- ParlayX client applications.
- CLI client applications.

---

# Operations

The following Billing operations are available:

`list`
>This operation lists all Balances available for this Account.
>
>Parameters:
>>*none*
>
>Results:
>`balance`
>>an array of available Balance names.

`credit`
>This operation adds funds to a Balance.
>
>Parameters:
>`balance`
>>a string: the Balance name. If the Balance with this name does not exist, it is created.
>
>`amount`
>>a non-negative number: the credited amount.
>
>`reference`
>>a string: arbitrary data used for references, dispute resolutions, etc.
>
>`description`
>>an object (usually a string or an array of strings): arbitrary data describing the transaction.

Results:
`amount`
>    a number: the current Balance value.


## remove

>    This operation removes a Balance.
>    **Note:** the Balance history is not removed. If the current Balance value is not zero, a record is added to the Balance history, changing the Balance value to zero.

>    Parameters:
>    `balance`
>    >    a string: the Balance name. The Balance with this name is removed.

>    Results:
>    `amount`
>    >    a number: the removed Balance value.


## reserve

>    This operation reserves funds from a Balance. The reserved amount cannot be used for other charges. The reservation can be released explicitly, returning funds to the Balance, or it can expire after the specified period of time.

>    Parameters:
>    `balance`
>    >    a string: the Balance name. This Balance must already exist.

>    `amount`
>    >    a number: the amount to reserve. If the reservation already exists, this value is used to increase the reserved amount.

>    `reserve`
>    >    a string, optional: if this parameter exists, it should specify the existing reservation. The `amount` will be added to this reserve. If this parameter is absent, a new reservation is created.

>    `expires`
>    >    a timestamp, optional: if this parameter exists, it specifies when this reservation should expire.

>    `overdraft`
>    >    optional: if this parameter exists, it is possible to reserve an amount larger than the current Balance value. If this parameter does not exist, attempts to reserve an amount larger than the current Balance value are rejected.

>    `charge`
>    >    a number: the amount to charge if this reservation expires and it is automatically released.

>    `reference, description`
>    >    these strings can be specified if the `charge` parameter is specified. They have the same meaning as the `credit` operation parameters, and they are used if this reservation is automatically charged when it expires.

>    Results:
>    `reserve`
>    >    a string: the reservation name (the same as the `reserve` parameter, if it was specified).

>    `amount`

a number: the current reservation value.

    `total`
        a number: the current Balance "free funds".


## release

This operation releases a reservation. The reserved amount is returned to the Balance.

Parameters:

`balance`
    a string: the Balance name. This Balance must already exist.

`reserve`
    a string: the existing reservation name.

Results:

`reserve`
    a string: the reservation name (the same as the `reserve` parameter, if it was specified).

`amount`
    a number: the released reservation value.

`total`
    a number: the current Balance "free funds".


## charge

This operation subtracts funds from a Balance or a reservation.

Parameters:

`balance`
    a string: the Balance name. This Balance must already exist.

`amount`
    a number: the amount to charge.

`reserve`
    a string, optional: if this parameter exists, the charge is applied to the already existing reservation. If this parameter is absent, the charge is applied to the Balance itself.

`overdraft`
    optional, cannot be specified together with the `reserve` parameter: if this parameter exists, it is possible to charge an amount larger than the current Balance value. If this parameter does not exist, attempts to charge an amount larger than the current Balance value are rejected.

`release`
    optional, cannot be specified without the `reserve` parameter: if this parameter exists, the reservation is released after the charge is made.

`reference`, `description`
    same as for the `credit` operation.

Results:

`amount`
    a number: the current Balance "free funds" or reservation value.

`total`
    a number: the current Balance "free funds" (only if the `reserve` parameter is specified).

expires

      a timestamp: the reservation expiration time (only if the `reserve` parameter is specified and the reservation has an expiration time set).

## read

      This operation reads the current Balance or reservation value.

      Parameters:

balance

      a string: the Balance name. This Balance must already exist.

reserve

      a string, optional: if this parameter exists, it must specify an existing reservation. The operation reads that reservation value. If this parameter is absent, the operation reads the Balance value.

      Results:

amount

      a number: the current Balance or reservation value.

expires

      a timestamp: the reservation expiration time (only if the `reserve` parameter is specified and the reservation has an expiration time set).

## history

      This operation reads the Balance transaction log.

      Parameters:

balance

      a string: the Balance name. This Balance must already exist.

timeFrom,timeTill

      timestamps: Balance records from recorded on or after `timeFrom` but before `timeTill` (exclusive) are read.

limit

      a number: the maximum number of records to display.
      If this number is positive, the newest records are returned first, and the oldest records are discarded if the record number limit is reached.
      If this number is negative, the oldest records are returned first, and the newest records are discarded if the record number limit is reached.

      Results:

history

      an array of transaction records. Each record contains the following elements:

Date

      a timestamp: the time of transaction.

amount

      a number: the transaction amount (positive for `credit` operations).

balance

      a number: the Balance value after this transaction.

reference,description

copies of the `credit` and `charge` operation parameters.

`calllog`
>This operation reads the Call Log files.

>Parameters:

>`timeFrom,timeTill`
>>timestamps: records for calls that took place on or after `timeFrom` but before `timeTill` (exclusive) are read.

>`limit`
>>a number: the maximum number of records to display.
>>If this number is positive, the newest records are returned first, and the oldest records are discarded if the record number limit is reached.
>>If this number is negative, the oldest records are returned first, and the newest records are discarded if the record number limit is reached.

>`filter`
>>optional string: if specified, the operation returns only the calls with peers whose addresses or "real names" match the specified string.

>Results:

>`callHistory`
>>an array of transaction records. Each record contains the following elements:

>>`Date`
>>>a timestamp: call start time (GMT).

>>`direction`
>>>the `"inp"` string for incoming calls, the `"out"` string for outgoing calls.

>>`To`
>>>the peer address string.

>>`RealName`
>>>(optional) the real name of the peer.

>>`Call-ID`
>>>the call Call-ID string.

>>`disconnected`
>>>a timestamp: call end time (GMT).

>>`connected`
>>>a timestamp: call connected time (GMT). If absent, the call has not been connected.

>>`dialogError`
>>>(optional) call completion/failure error code string.

>>`dialogApp`
>>>(optional) if a call has been answered by a PBX application, the application program name string.

>`amount`
>>the total number of found call records

**CommuniGate Pro**

**WebMail** | **Mailboxes** | **Messages** | **Composing** | **Contacts** | **Calendar** | **Tasks** | **Files** | **Notes** | **S/MIME**

# WebMail: WebUser Interface

- **WebUser Interface Pages**
- **WebUser Interface Login**
- **Browser Authentication Login**
- **WML/IMode Login**
- **WebUser Interface Auto-Signup**
- **WebUser Interface Settings**
- **Password Modification**
- **Public Info Editor**
- **Automated Rules**
- **RPOP Accounts**
- **Trash Management**
- **Secure Mail (S/MIME)**
- **Access Rights Management**

The CommuniGate Pro Server provides Web (HTTP/HTML) access to user accounts. The WebUser component works via the HTTP module and allows users to read and compose messages and to perform account and Mailbox management tasks using any Web browser.

Even if you prefer a regular POP or IMAP mail client, the WebUser Interface can be used to access the features unavailable in some mailers. For example, the WebUser Interface can be used to specify Subscriptions and Access Control Lists for account Mailboxes - the features many IMAP clients do not support yet.

If the WebCal Service is enabled for your Account, you can use the WebUser Interface to create Events (Meetings and Appointments) and ToDo items (Tasks), to accept and cancel them, to view your Calendar and ToDo lists.

The WebUser Interface is completely customizable. The CommuniGate Pro package includes several "stock" Skins - an unnamed one (it uses a minimal set of graphic elements and it does not use any scripting) and several named Skins. Named Skins usually provide more graphic-intense interfaces. Each CommuniGate Pro installation can use an unlimited number of custom Skins.

All pages shown in this section are displayed using the simple unnamed stock Skin. The same pages may look differently when displayed with other Skins.

# WebUser Interface Pages

The WebUser Interface consists of several types of HTML pages that you can access using the controls - links and buttons. When you access the server using a WAP/WML (wireless phone) or IMode browser, WML or IMode pages with the same functionality are generated and sent to your wireless device.

Hello page
> This page is displayed when you log into the system. It allows you to switch to other WebUser Interface pages, as well as to other portions of your site.

Mailboxes page
> This page lists all Mailboxes in your account and allows you to create, rename, and remove Mailboxes, and to open Mailboxes so you can browse the messages stored in your Mailboxes. See the Mailboxes section for more details.

Mailbox page
> This page lists all messages stored in the selected Mailbox. You can copy, move, redirect, forward and delete listed messages. You can open and read messages listed on the Mailbox page. See the Mailboxes section for more details.

Message page
> This page presents the content of the selected message. You can read the message, copy, move, delete, redirect, and forward the open message, and you can reply to it. See the Messages section for more details.

**Compose page**
> This page allows you to compose a new message, and send it. It can also be used to create and modify Notes, and calendaring (Event and ToDo) items. See the Composing section for more details.

**Settings pages**
> These pages allow you to customize your WebUser Interface.

**Files page**
> This page allows you to manage your File Storage.

**Contacts pages**
> These pages allow you to browse your Contact-type (AddressBook-type) Mailboxes ("folders"), and to edit your Contact and Contact Group items. See the Contacts section for more details.

**Notes page**
> This page allows you to manage your Notes. See the Notes section for more details.

**Calendar and Tasks pages**
> These pages allow you to browse your Calendar-type and ToDo-type Mailboxes ("folders"). See the Calendar and Tasks sections for more details.

---

# WebUser Interface Login

The Login page allows you to log into the WebUser Interface by presenting your user name and password:

<div align="center">

**Welcome to CommuniGate Pro,
the domain1.dom Communication Server!**

</div>

| | | |
|---|---|---|
| **Auto-Login** | | **Mailing Lists** |
| | | **Directory** |
| **Login Name** | | **Signup** |
| **Password** | | **Security Certificate** |
| **Layout** | | **S/MIME Certificate** |
| 64.173.55.169 | Disable Fixed Address Check | **Mail to Postmaster** |
| 08:44:56 | Disable Cookie check | |

*Pronto!*

Check that the Domain Name (domain1.dom in this example) correctly specifies the Domain your Account belongs to. If you cannot open the Login page of the proper Domain, you still can log in, but then you would need to enter your full Account name (in the *accountName@domainName* form).

Your Account WebUser Preferences (Settings) may enable some additional security mechanisms, such as the Fixed IP Address mechanism and/or the Cookies mechanism.
When you connect using a network with multi-home proxies (such as the AOL network), your requests come to the CommuniGate Pro server from different network addresses, even when you continue to use the same browser on the same network. If you have to connect to the Server from such a network, you may want to disable the Network Address feature for this session, otherwise you will get disconnected very quickly. Some browsers do not support "cookies". If you have to connect to the Server from such a browser, you may want to disable the Cookie check for this session.
If you always connect from a proxied network or if you always use a browser that does not support cookies, you may want to disable this security options in your Account WebUser settings, so you won't have to disable them manually every time you try to log in.

---

# Browser Authentication Login

You can use your browser Authentication capabilities as an alternative method to log into the WebUser Interface. Click the Auto-Login link on

the login page, or point your browser to `http://yourserver:port/login/`

Your browser may display a dialog box asking you to supply your credentials, or it will use your already activated credentials. The browser then sends a request with these credentials.
When the Server verifies and accepts your credentials, a WebUser Session is created and your browser is redirected into that session.

Browser Authentication is more secure if a session is established via a clear-text link, and secure HTTP Authentication methods are enabled and supported in the browser.

Browser Authentication is convenient because the browser remembers the supplied credentials and it will login automatically the next time you direct it to this special URL. If the browser supports the Single Sign-On Authentication (such as GSSAPI/Kerberos and/or client Certificates), this method allows you to log into the WebUser Interface without supplying your credentials again.

**Note:** it is not recommended to use Browser Authentication on workstations shared by several people (because the browser remembers the supplied credentials), unless the workstation and the Server are configured to use Single Sign-on Authentication (and thus the supplied credentials are always the actual credentials of the current workstation user).

---

# WML/IMode Login

The CommuniGate Pro Server automatically detects WML and IMode devices, and automatically switches to the WML or IMode interface when these devices are used.

If the Server does not correctly detects your wireless device type, you may want to specify the markup language explicitly.

To use the WML interface, point your device browser to `http://yourserver:port/wml`

To use the European version of the IMode interface, point your device browser to `http://yourserver:port/imode`

To use the Japanese version of the IMode interface, point your device browser to `http://yourserver:port/imodejp`

To use the regular HTML interface, point your device browser to `http://yourserver:port/html`

---

# WebUser Interface Auto-Signup

The Administrator of your Server or your Domain can enable an auto-signup feature. If this feature is enabled, you can open the Signup page (using a link on the Login page), enter your Account name ("username"), password and some other parameters, and create a new Account in this Domain. The system then logs you in that Account:

<div align="center">

**Welcome to CommuniGate Pro,
the domain1.dom Messaging Server!**

</div>

| Login Name | Real Name |
|---|---|
| Password | Reenter Password |
| Layout | |
| | |
| Forgotten Password Recovery | E-Mail Password to |

The passwords entered in both passwords fields should match.

You may want to enter the Forgotten Password Retrieval E-mail address - the system will send your password there if you forget it. Please note that:

- This E-mail address should not be the address of the Account you are creating, because when you forget your password, you cannot open that Account and read the message with your recovered password. This E-mail should be the E-mail Address of an account you have with some different ISP.
- The Server will be able to send your password via E-mail, only if your password is stored in the clear-text form or it can be decoded by the Server. If the password stored as one-way encoded "hash", the Server will not be able to decode it, and the Server will not be able to send you the decoded password string.

# WebUser Interface Settings

You can tune the WebUser Interface by modifying settings on the Settings pages.

The Settings pages contain options that customize [Accesses to Mailboxes](#), [Mailbox Browsing](#), [Message Browsing](#), and [Message Composing](#).

These pages also contain some generic settings:

- The Interface ("Skin") to use.
- The security options used to protect your WebUser sessions.
- The character set options.

The WebUser Interface Settings page contains a link to the Account [Mailbox Subscription](#) and [Mailbox Aliases](#) page.

| | |
|---:|:---|
| Layout: | default(***) |
| Language: | default(English) |
| Time Zone: | default(HostOS) |
| Require Fixed Network Address: | No |
| Use Cookies: | Yes |

Layout
> Use this setting to specify the Layout ("Skin") of the WebUser Interface. Select the `Basic` option to use the default ("unnamed") Skin. When you change the Layout setting, you need to Logout and Login again to use the newly selected Skin.

Language
> Use this setting to specify the Language to use on the selected Skin. The menu shows all Languages available for the selected Skin.

Require Fixed Network Address
> Select this option to enable the Network Address security check. When this option is enabled, the Server remembers the Network Address you logged in from, and then all HTTP requests to the established WebUser Interface session must come from the same Network Address, otherwise requests are rejected.
> **Note:** Do NOT select this option if you plan to connect from the AOL network or other networks that use outgoing HTTP proxies.
> **Note:** You can disable the Network Address security check when you [login](#).

Use Cookies
> Select this option to enable HTTP "cookie"-based security check. When this option is enabled, some "cookie" information is sent to your browser when you login, and the browser resends that information back to the CommuniGate Pro server every time you access a WebUser Interface session page. Other browsers cannot access your WebUser Interface session even if they connect from the same Network Address, as they do not posses the proper "cookie" information.
> **Note:** Do NOT select this option if you plan to use browsers that do not support "cookies".
> **Note:** You can disable the "cookie" security check when you [login](#).

Time Zone
> Use this setting to select the time zone you are working in. The WebUser Interface and XIMSS clients will use the select time zone to show the date and time values.
> Only of your time zone is not listed, Select the `HostOS` value to use the Server OS time zone.

| Text Encoding | |
|---:|:---|
| Preferred Character Set: | default(Western European (ISO)) |
| Use Unicode (UTF-8) for: | Reading and Composing |

Preferred Character Set
> Use this setting to select the character set you use most. New messages you compose will be encoded using the selected Preferred Charset. If a message does not have the charset specified, it is displayed using the Preferred Charset.

Use Unicode (UTF-8) for
> Use this setting to specify how your browser can utilize the Unicode (UTF-8) encoding. Select the Reading and Composing option if you use a modern browser.

# Password Modification

The WebUser Interface Settings page contains a link that opens the Password Editor page:

| Settings: Password | | General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info |
|---|---|---|
| **Password Modification** | Current Password | |
| | New Password | |
| | Reenter Password | |
| **Forgotten Password Recovery** | E-Mail Password to | |
| | | |

To update your password, enter your current password, then enter your new password twice, and click the Modify button.
**Note:** The New Password fields may be absent. This means that the Administrator did not allow you to modify your Account password.

You can specify the Password Recovery E-mail address. It should be an address of some other account, most likely - on some other system. If you ever forget your CommuniGate Pro Account password, you will be able to ask the Server to send your password to that E-mail address. To set the Password Recovery E-mail address, enter your current password and the Password Recovery E-mail address into their fields and click the Modify button.

---

# Public Info Editor

The Public Info Editor page can be used to modify your Account data stored in the Directory. Other users can retrieve this information from the Directory using any LDAP client, or using WebUser Interface to the CommuniGate Pro Directory.

The System Administrator defines the set of Account Settings used as user's Public Info. If the Public Info Editor page contains no fields, the System Administrator has not specified any Public Info settings.

| Settings: Public Info | | General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info |
|---|---|---|
| | Work Phone | |
| | City | |
| | | |

You can update the Public Info Account Settings by modifying the data in the value fields. Please note that the CommuniGate Pro Server can 'rename' Account Settings when storing them in the Directory. The `City` setting may be stored as the standard `l` directory attribute, while the `Work Phone` setting can be stored as the `telephoneNumber` attribute in your Directory record. The Server Administrator specifies the Account Settings <-> Directory Attribute renaming rules.

You can modify your "profile" information (the `profile.vcf` vCard file) using the Profile Editor:

| | My Profile vCard | |
|---|---|---|
| | File As | |
| | E-mail | |
| | | |
| | Web Site | |
| **Full Name** | Title | |
| | First Name | |
| | Middle Name | |
| | Last Name | |

| | | |
|---|---|---|
| **Telephone** | Suffix | |
| | Work | |
| | Home | |
| | Cell | |
| | Fax | |
| | Assistant | |
| | | |
| | Organization | |
| | | |
| | Job Title | |
| | Profession | |
| | Photo | no file selected |

Enter your personal data you want to disclose in your "profile" and click the Save Contact button. You can attach this profile information to E-mail messages you sent.

This profile data is used with your Instant Message clients.

---

# Automated Rules

The WebUser Interface provides access to the Account [Automated Mail Processing Rules](#). If the `Can Modify Account Rules` [Account Setting](#) is not enabled, then you can view the Account Mail Rules, but you cannot modify them.

You can turn the Auto-Reply option on and you can modify the Auto-Reply message text even if your Account does not have a right to modify any Mail Rules.

See the [Automated Mail Processing Rules](#) section to learn how to specify the Rules.

The WebUser Interface provides access to the Account [Automated Signal Processing Rules](#). If the `Can Modify Signal Rules` [account option](#) is not enabled, then you can view the account Signal (Call) Rules, but you cannot modify them.

---

# RPOP Accounts

The WebUser Interface provides access to the list of the [Remote POP Accounts](#) that the system polls on your behalf.

If the `Can Modify RPOP Accounts` [account option](#) is not enabled, then you can view the list of RPOP Accounts, but you cannot modify them.

See the [RPOP Module](#) section to learn how to specify the Remote POP Accounts to poll.

---

# Trash Management

The WebUSer Interface Settings allow you to specify how the delete operations are handled:

| Trash Management | |
|---|---|
| Message Delete Method: | default(Move To Trash) |
| Trash Folder: | default(Trash Can) |
| Keep Message Received Time: | default(No) |
| On Logout Remove from Trash if Older than: | 2 days |
| Junk Folder: | default(Junk) |

`Message Delete Method`

Set this option to Immediately if you want to permanently remove a message when you click the Delete link or button.

Set this option to Move To Trash if you want to move deleted messages to the special Trash Mailbox, so they can be recovered from there.

Set this option to Mark if you the Delete operation to mark messages as "deleted", without actually removing them. Then you can use the Purge Deleted operation to remove all Mailbox messages marked as Deleted.

The remaining options can be used when the Move To Trash method is selected.

`Trash Mailbox`

This setting allows you to specify the Mailbox to be used as Trash. If you access your account with some other mailer that uses a Trash Mailbox, too, you may want to configure the WebUser Interface to use the same Mailbox as Trash. For example, the Microsoft Outlook client uses the `Deleted Items` Mailbox as a Trash Mailbox.

`Keep Message Received Time`

If this option is enabled, then messages moved to Trash keep the Received ("Internal") time attribute, it shows the time when the message was received. If this option is disabled, the Received time attribute for messages moved to Trash is changed to the time when they were moved. This option has an effect on the next option.

`On Logout Remove from Trash if Older than`

When you end your WebUser Interface or [XIMSS](#) session, the Server checks the Received date of the messages in the Trash Mailbox, and removes all messages older than the specified period of time. Depending on the Keep Message Received Time option value, it allows you to keep only *recent* messages in the Trash, or to keep only *recently deleted* messages in the Trash.

`Junk Mailbox`

This setting allows you to specify the Mailbox to store Junk mail.

`On Logout Remove from Junk if Older than`

When you end your WebUser Interface or [XIMSS](#) session, the Server checks the Received date of the messages in the Junk Mailbox, and removes all messages older than the specified period of time.

# Secure Mail (S/MIME)

The WebUser Interface allows you to send and receive digitally signed and encrypted messages. It can decrypt encrypted messages, verify digital signatures, and perform other Secure Mail operations.

The Secure Mail functionality is available only if your Account and Domain have the S/MIME [Service](#) enabled.

See the [WebUser Interface S/MIME](#) section to learn how to use Secure Mail functions.

# Access Rights Management

The WebUser Interface allows you to set and modify your [Account Access Rights](#).

Open the [Mailboxes Settings](#) page and locate the Access Rights panel:

| Management | | | | | |
|---|---|---|---|---|---|
| **Access Control List** | Identifier | Delegate | Call Control | Admin | Create |
| | | | | | |

**WebMail** | **Mailboxes** | **Messages** | **Composing** | **Contacts** | **Calendar** | **Tasks** | **Files** | **Notes** | **S/MIME**

# WebMail: Mailboxes

- **Access to Mailboxes**
- **Mailbox Browsing**
- **Mailbox Management**
- **Mailbox Subscription Management**
- **Mailbox Aliases Management**
- **Access to Mailboxes by Name**

The CommuniGate Pro WebUser Interface provides access to user Mailboxes or "folders". You can display the list of Mailboxes in your Account, create new Mailboxes, rename and remove Mailboxes, open and view Mailbox, search Mailboxes for certain data, etc.

Mailboxes can be of a regular type - they contain E-mail messages. INBOX is a regular type Mailbox that contains all messages received by your Account.

You can also create Contacts-type (AddressBook-type) Mailboxes (folders), Notes-type Mailboxes, and (if the Calendaring Service is enabled for your Account) the Calendar-type and Tasks-type Mailboxes.

# Access to Mailboxes

Click the Folders link to open the Mailboxes page. This page displays your Mailboxes, and it allows you to open an existing Mailbox or to create a new one:

| Management | | | | | |
|---|---|---|---|---|---|
| **Mailbox** | | | | | |

| **Folders** | | | | used 238K of 3M | |
|---|---|---|---|---|---|
| Filter: | | | | 3 selected | |
| **Folder** | **Size** | **Messages** | **New** | **Unread** | |
| INBOX | 20K | 5 | **2** | 3 | |
| Friends | 200K | 56 | **4** | 4 | |
| Friends/Family | 18K | 3 | | | |
| 175K in Trash | | | | | |

To open a Mailbox, click on its name.

Some Mailboxes are "unselectable" and their names are not URL links.

You can open the Settings page and specify which Mailboxes should be displayed in the Mailboxes page:

| **Settings: Folders** | General   Password   Folders   Compose   Calendar   Contacts   Secure Mail   Public Info |
|---|---|
| **Folder List Viewer** | All Account Folders          default(Yes) |
| | All Subscribed Folders          default(Yes) |

Display All Account Mailboxes
>    If this option is selected, all Mailboxes and Mailbox aliases created in your Account are listed.

Display Subscribed Mailboxes
>    If this option is selected, the Mailboxes page lists all Mailboxes your Account is [subscribed](#) to (including foreign Mailboxes).
>    If this option is selected, the newly created Mailboxes will be automatically added to the subscription list.


To create a Mailbox, select the Mailbox type (regular Mailbox, AddressBook, Calendar, etc.) and type in the new Mailbox name, then click the Create button.

---

# Mailbox Browsing

You can browse a Mailbox by clicking its name (link) on the [Mailboxes](#) page. A Mailbox page displays the messages stored in the Mailbox, it provides checkboxes to select messages, and the controls for performing operations on the selected messages:

| INBOX | | | | | 23 of 1238 unread |
|---|---|---|---|---|---|
| 20 | | Filter: | Search: | | **1 selected** |
| **Status** | **From** | **Subject** | | **Size** | **Received** |
| ✉ | TestList administration | Welcome! | | 871 | 25-Nov-06 |
| ✉ | Technical Support | Fwd: [*] CommuniGate Pro 5.1.3 released | | 4K | 25-Nov-06 |
| ✉ | Technical Support | TEST - text & 2 gifs | | 11K | 25-Nov-06 |
| ✉ | Technical Support | Fwd: TEST - text & 2 gifs | | 13K | 25-Nov-06 |
| ✉ | philip@node5.communigate.com | (no subject) | | 5K | 25-Nov-06 |
| ✉ | Technical Support | FWD:(no subject) | | 5K | 25-Nov-06 |
| ✉ | Technical Support | Fwd: HTML letter (alternative) (no subj | | 6K | 25-Nov-06 |
| ✉ | John R. Smith | Re: Weird Problems | | 2015 | 03-Dec-06 |
| ® | Douglas M. | bad log file from our server | | 8K | 11-Dec-06 |
| ✗ | U&B | LDAP | | 1575 | 21-Dec-99 |
| ✉▶ | James Green | design suggestion | | 3K | 23-Dec-06 |
| ✉▶ | Adam Drake | Transmission problems between SIMS and | | 4K | 05-Jan-07 |
| | ✉ | | | | |
| | ⚑ | | | | |
| | ✗ | | | | |
| | **Folder Management** | | | | |

For each message in the Mailbox, several message header fields are displayed. Messages are sorted by the highlighted field. Click the field name to highlight a different field and to change the sorting order.

A message can be opened using a link in the first or highlighted column.

Display
>    This button tells the WebUser module to display not more than the specified number of the Mailbox messages. If the Filter field is not empty, only the messages with the highlighted field containing the filter string are displayed. If the Search field is not empty, only the messages containing the search string are displayed.

Read
>    The Set button can be used to mark the selected messages as "read", the Clear button can be used to mark the selected messages as "unread".

Flagged
>    The Set button can be used to mark the selected messages with a flag, the Clear button can be used to remove the flag marker from the selected messages.

Copy To
>    This button can be used to copy the selected messages into the specified Mailbox.

Move To
>    This button can be used to copy the selected messages into the specified Mailbox; the original message is deleted or it is marked as

`deleted` (if the WebUser Interface Delete Mode is set to `Marked`).

Redirect To, Forward To
> This button can be used to redirect or forward the selected messages to the specified addresses. The address field below the buttons should contain one or several addresses separated with the comma signs.
> The To/Cc fields of the selected messages are replaced with the specified address(es), unless you prefix the address list with the `[bcc]` string.

Mailbox Management
> This link is used to open the [Mailbox Management](#) page.

The following buttons appear if the WebUser Interface Delete Mode is set to `Mark`:

Delete
> The Set button is used to mark the selected messages as "deleted", the Clear button can be used to clear the "deleted" markers.

Purge Deleted
> This button is used to remove the messages marked as "deleted" from the Mailbox.

If the the WebUser Interface Delete Mode is set to `Via Trash` or `Immediately`, the following button appears:

Delete
> Click this button to move the selected message(s) to the Trash Mailbox (the `Via Trash` Delete Mode) or to mark all selected messages as "deleted" and remove all marked messages immediately (the `Immediately` Delete Mode).

You can open the Settings page and specify how Mailboxes should be displayed:

| Folder Viewer | | | | | | | |
|---|---|---|---|---|---|---|---|
| Display: 20 | | | | Refresh Every: 60 sec | | | |
| Fields: Custom | Status | From | Subject | Size | Received | Sent | Reverse |
| Sort: | | | | | | | Yes |

Display
> This option specifies how many messages should be displayed on one Mailbox page. If a Mailbox has more messages than this option specifies, arrow links appear to allow you to "page" the entire Mailbox.

Refresh Every
> This option specifies how often the Mailbox pages should be automatically updated.

Fields
> This set of options specifies the message fields to be displayed in the Mailbox pages. Select an empty option and click the Update button to remove a field.
> **Note:** the `From` field column displays the `To` field data when you open Mailboxes assigned to keep your Sent or Draft messages, or their sub-mailboxes.

Sort
> This set of radio buttons allows you to select the field for initial (default) Mailbox sorting.

Reverse
> This option specifies the initial (default) Mailbox sorting order.

The WebUser Interface Settings page also allows you to specify the Delete Mode:

| Trash Management |
|---|
| Message Delete Method: Move To Trash |

Move To Trash
> The delete operation moves the selected message(s) to the Trash Mailbox. This option available for multi-mailbox Accounts only. If the Trash Mailbox does not exist, the first delete operation creates it.

Mark
> The delete operation marks the selected message(s) as "deleted". The marked messages can be removed using the Purge Deleted operation.

Immediately

The delete operation marks the selected message(s) as "deleted" and then immediately deletes all marked messages from the Mailbox.

When you click a Calendar-type Mailbox name, the [Calendar View](#) page is opened.

When you click a Tasks-type Mailbox name, the [Task List View](#) page is opened.

When you click a AddressBook-type Mailbox name, the [Contacts List View](#) page is opened.

When you click a Note-type Mailbox name, the [Notes List View](#) page is opened.

# Mailbox Management

The Mailbox Management page allows you to set the ACL ([Access Control List](#)) settings for the selected Mailbox, to rename, and to remove the Mailbox.

To rename a Mailbox, type the new Mailbox name into the New Folder Name field and click the Rename Folder button. If the Rename Sub-Folders option is selected, all submailboxes of this Mailbox will be renamed, too. If you are renaming the Mailbox `Sent` into `Sent in 2000`, and you also have the `Sent/customers` submailbox, that submailbox is renamed into `Sent in 2000/customers` if the Rename Sub-Folders option is selected.

**Note:** If you rename your `INBOX`, the new empty INBOX is automatically created.

To remove a Mailbox, click the Remove Folder button. If the Remove Sub-Folders option is selected, all submailboxes of this Mailbox will be removed, too. If you are removing the Mailbox `Sent`, and you also have the `Sent/customers` submailbox, that submailbox is removed, too - if the Remove Sub-Folders option is selected.

**Note:** You cannot remove your `INBOX`.

To grant Mailbox access rights to a user, enter the user name into the Identifier field, select the desired access rights, and click the Update button. To grant an access right to everybody, use the word `anyone`. To remove certain rights from a particular user, "grant" those rights to the identifier `-username`. See the [Mailboxes](#) section for more details.

| **INBOX: Folder Management** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Access Control List** | | | | | | | | | |
| **Identifier** | **Lookup** | **Select** | **Seen** | **Flags** | **Insert** | **Post** | **Create** | **Delete** | **Admin** |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| Visible to Mobile Devices | | | | | | | | | |

`Visible to Mobile Devices`
   If this option is not selected, [AirSync](#) devices will not see this Mailbox. You may want to disable this option, if your mobile devices cannot handle all of your Mailboxes.

If you select the Apply to Sub-Folders option, then the selected Access Control List and the Visible to Mobile Devices option will be assigned not only to the current Mailbox, but to all its sub-mailboxes.

# Mailbox Subscription Management

The Mailboxes Settings page allows you to set the [Mailbox Subscription](#) - the list of your own and foreign Mailboxes you want to use.

You can open the Mailboxes Settings page using the link on the Settings page:

| **Management** |
|---|

Type a Mailbox name into an empty field and click the Update button to add a Mailbox to the subscription list.

To specify a Foreign Mailbox, type the Tilda sign (~), the user name, the slash sign (/) and then the Mailbox name. Make sure that user has already granted you the Select access right for that Mailbox.

# Mailbox Aliases Management

The Mailboxes Settings page allows you to set the Mailbox Aliases - the list of simple names for foreign Mailboxes. You should use Mailbox aliases if you want to access foreign Mailboxes via IMAP clients that do not support the foreign Mailboxes concept.
It is not recommended to use Mailbox aliases with more advanced IMAP clients or with the WebUser Interface itself, since they add unnecessary complexity to Mailbox management.

| Management | | |
| --- | --- | --- |
| **Folder Aliases** | Alias Name | Folder Name |
| | | |

Type a simple Mailbox name into an empty field in the left column, type the name of a foreign Mailbox into the right column field, and click the Update button to create a Mailbox alias.

Your mail client software will list the created Mailbox aliases as well as the real Mailboxes created in your Account. You can open a Mailbox alias as you open a real Mailbox, and the specified foreign Mailbox will be opened.

Change the Mailbox alias name to an empty string and click the Update button to remove the Mailbox alias.

# Access to Mailboxes by Name

You may want to open a foreign Mailbox without including it into your subscription list and without creating a Mailbox alias.

Open the Mailboxes Settings page and type the Mailbox name in the Open Mailbox panel and click the Open button:

| Management | |
| --- | --- |
| **Open Folder** | |

| Introduction |
| Installation |
| SysAdmin |
| Network |
| Objects |
| Storage |
| E-mail |
| Real-Time |
| Access |
| Services |
| Directory |
| Clusters |
| Applications |
| Miscellaneous |
| Licensing |
| WebMail |
| Pronto! |
| PBX |

WebMail  Mailboxes  **Messages**  Composing  Contacts  Calendar  Tasks  Files  Notes  S/MIME

# WebMail: Messages

- **Message Browsing**
- **Storing Addresses**
- **Messages Copying**
- **Messages Redirecting**
- **Storing and Removing Attachments**

The CommuniGate Pro WebUser Interface allows you to read messages stored in your Account Mailbox(es). To read ("open") a message, open the [Mailbox](#) page first and click on the message link.

## Message Browsing

The WebUser Interface allows you to view messages in your Mailboxes. It checks the MIME structure of a message and decodes its MIME parts.

A Mailbox message is displayed as an HTML page, containing the important fields of the message header, the decoded message body, and the controls. Text, HTML, and graphics MIME parts are displayed, other parts (attachments) are shown as icon links that allow you to download these parts.

The `multipart`-messages are displayed according to the MIME multipart rules, and the nested messages (forwarded messages, reports, digests) are displayed, too.

The message header (and message headers of all embedded messages) has icon-links that allow you to view the complete header information, and to view the undecoded message body.

You can use the following controls (links/buttons) on Message pages:

Next Unread
    Click this control to open the next unread message (a messages without the `read` marker) in the Mailbox.

Back to *mailboxname*
    Click this control to close the message and to open the Mailbox page.

Close as Unread
    Click this control to mark this message as unread (removes the `read` marker), to close the message page, and to open the Mailbox page.

Delete
    Click this control to mark this message as deleted (sets the `deleted` marker), to close the message, and to open the Mailbox page.

Undelete
    Click this control to remove the `deleted` marker from the message.

Reply

Click this control to open the [Compose page](#) and to send a reply message.

Reply To All
>    Click this control to open the [Compose page](#) and to send a reply message. The pre-composed recipients list will include not only the author of the original message, but all message Cc: and To: recipients, too.

Forward
>    Click this control to open the [Compose page](#) and to forward a message.

Set Flag
>    Click this control to add the `flag` marker to the message.

Reset Flag
>    Click this control to remove the `flag` marker from the message.

Redirect
>    Click this control to redirect the message to the specified address(es). If you need to type in several addresses, separate them with the comma sign.

Edit Draft
>    This control appears only for *draft* messages; click this control to open the message in the [Compose page](#), so you can complete it and send it to its recipients.

You can open the Settings page and specify how messages should be displayed:

| **Message Viewer** | | | |
|---|---|---|---|
| Show HTML: | default(inline) | Use Message Charset: | default(No) |
| Send Read Receipts: | default(Automatically) | Display Links: | default(safely) |
| Show Images: | default(safely) | | |
| Fields: | Custom | | |

Show HTML
>    This option specifies how the WebUser Interface should process messages in the HTML format:

>    in frame
>    >    In this mode the HTML portions of messages are displayed in an *embedded frame*, providing complete separation of the message HTML code from the WebUser Interface Message page code. If you use a browser that does not support embedded frames, you will have to click a special link to open the HTML portion of the message in a separate window.

>    inline
>    >    In this mode the HTML portions of messages are inserted into the WebUser Interface Message page code. The WebUser Interface checks the message HTML portion code and removes some tags that may distort the entire WebUser Message page.

>    disabled
>    >    In this mode the plain text messages portions are displayed instead of the HTML portions. If only an HTML portion exists, it is displayed as a plain Text, after all HTML tags are removed from it.

Show Links
>    This option specifies how the WebUser Interface should display links in HTML and plain-text messages:

>    directly
>    >    In this mode links are displayed "as is". If the WebUser protection methods (Fixed IP Address, Cookies) are disabled, attackers can receive your WebUser Session ID string when you follow links they have sent to you, and you open a page on the attackers' Web site.

>    safely
>    >    In this mode displayed links are modified, so the referenced Web sites do not receive the "referrer" information containing your WebUser Session ID. If your browser does not send the "Referer" field or your firewall remove these fields from your requests, this option will not work, and you should select the `directly` mode.

>    disabled
>    >    In this mode displayed links cannot be used as links.

Show Images

This option specifies how the WebUser Interface should process references to external images and other objects:

`directly`
> In this mode image references are used "as is". If the WebUser protection methods (Fixed IP Address, Cookies) are disabled, attackers can receive your WebUser Session ID string when you view an image object on the attackers' Web site.

`safely`
> In this mode object references are modified. The Web sites hosting external images do not receive the "referrer" information containing your WebUser Session ID. This option does not work with some browsers.

`proxy`
> In this mode object references are modified to start a "proxy" application on the Server that retrieves the referred object and sends it to the client. The Web sites hosting external images do not receive the "referrer" information containing your WebUser Session ID. Your Account must have the `HTTP` Service enabled.

`disabled`
> In this mode image references are removed, and no external image is displayed.

This option has no effect on "internal" images, i.e. the images sent within the same message.

`Use Message Charset`
> If you have disabled the Unicode (UTF-8) display because your browser does not support it well, you can meet problems viewing messages are composed using the charsets incompatible with your preferred charset.
> If you set this setting to Yes, the message viewer page will be displayed in the charset that allows you to read the message content, but the page design (navigation links, etc.) can become unreadable if it uses national characters.
> If you set this setting to No, the message page will be always displayed correctly, but the message content can be unreadable if the message is composed using an incompatible charset.
> If you have allowed your browser to use the UTF-8 charset for Reading, this setting has no effect.

`Send Read Receipts`
> This option specifies what happens when you open a message for which the message author has requested a "read notification".

`disabled`
> Requests for Read Notifications are ignored.

`manually`
> When you open a message that contains a Read Notification request, a "Send Confirmation" button appears, allows you to send a confirmation (a Read Receipt).

`automatically`
> When you open a message with a Read Notification request for the first time, a confirmation (Read Receipt) message is automatically sent to the message author.
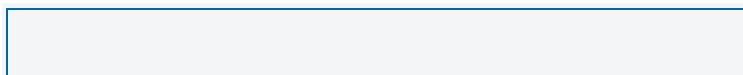
`Fields`
> The fields listed in this set are displayed in the message headers.

---

# Storing Addresses

The WebUser Interface allows you to store the sender's name and E-mail address in your Address Book. Click the Take Address button to add a the message `From:` address to your Address Book:

[addressbook]

If a message contains a *digital signature* and your CommuniGate Pro Account has the Secure Mail feature enabled, you can add the sender's *certificate* (also known as *digital ID*) to your Address Book:

See the [Secure Mail](#) section of the manual for more details.

# Message Copying

You can copy or move the opened message to any other Mailbox in your Account (or to a Mailbox in any other Account if you have the proper access rights).

<div style="border:1px solid #888; padding:40px; text-align:center;">Proposals</div>
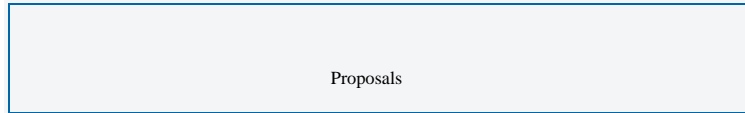
Select the Mailbox you want to copy to, and click the Copy to button.

If you click the Move To button, the message will be copied, and the original will be either removed, or marked as `deleted` (if the WebUser Interface Delete Mode is set to `Marked`).

# Message Redirecting

You can redirect the opened message to one or several E-mail addresses.

<div style="border:1px solid #888; padding:60px;"></div>
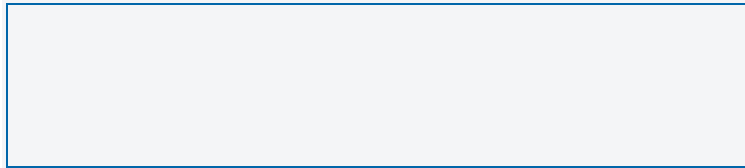
Type the addresses you want to redirect the message to, separating them with the comma (,) sign, and click the Redirect button.

# Storing and Removing Attachments

The WebUser Interface allows you to store message attachments on your own computer by just clicking the attachment name in your browser. If the File Storage feature is enabled for your CommuniGate Pro Account, you can store attachments and embedded images directly to your File Storage folders.

When a message has file parts, the Store In button appears:

<div style="border:1px solid #888; padding:50px; text-align:center;">My Public Site</div>

Select a File Storage folder, and click the Store File button. All message attachments and images will be copied into the selected folder.

Click the Remove File button to remove message attachments. The original message is deleted or moved to your Trash folder (depending on your settings), and a message copy with removed attachments is stored in your Mailbox.

# WebMail: Composing Messages

- **Opening the Composer Page**
- **Composer Settings**
- **Replying to Messages**
- **Forwarding Messages**
- **Attaching Files**
- **Attaching Your Contact Data**
- **Checking Spelling**
- **Delivery Status Notification**
- **Message Disposition Notification**
- **Address Book**

CommuniGate Pro WebUser Interface allows you to compose E-mail messages and send them to one or several recipients - local and remote. You can use Address Book(s) to select message recipients.

Composed messages can contain one or several file attachments.

Composed messages can be saved as *drafts*, without actually sending them. Draft messages can be opened later, completed, and sent.

All sent messages can be stored in a designated Mailbox.

## Opening the Composer Page

The Composer page can be opened either directly, by clicking on the New Mail link, or as a result of a Reply or Forward command.

The Composer Page contains the following panels:

- The message header panel with the `To`, `Subject`, `To`, and `Bcc` fields, and option controls.
- The message body text area.
- The attachment controls.

To send a message, fill the header panel fields, type the message text into the message body text area, and click the Send button.

## Composer Settings

Each message you send using the CommuniGate Pro WebUser Interface contains your address as the message `From` address, and it can also contain your signature.

Use the Settings page to specify the options that apply to all messages you compose and send using the CommuniGate Pro WebUser Interface.

| From Address | | |
|---|---|---|
| Name | Address | |
|  |  |  |

These fields allows you to specify the `From:` addresses for the messages you send using the WebUser Interface. By default, the address is set to the name of your Account on this Server and the 'Real Name' attribute of your Account. The Domain Administrator may apply some restrictions to the Name and Address you can choose.

| Message Composer | | | |
|---|---|---|---|
| Auto Wrap: | default(Flowed) | Text columns: | 72 |
| Reply Quoting: | default(>) | MIME-encode Headers: | default(Yes) |
| Signature: | | | |

Text columns
> This option specifies the width of the Composer field you use to enter your message texts.

Auto Wrap
> This option specifies if the composed text should be hard-wrapped (cut into individual lines) before sending. If you disable this feature, some recipients may see entire paragraphs of your messages as single, very long lines.

MIME-encode Headers
> If this option is set, message header fields containing non-ASCII (national) symbols are sent using MIME encoding.

Signature
> This text is automatically added to all messages you compose using the WebUser Interface.

Store Sent Messages in
> If this option is selected, a copy of all messages you compose using the WebUser Interface is stored in the specified Mailbox.

Store Drafts in
> If this option is selected, you can save partially composed messages in the selected Mailbox. Later you can open these *draft* messages, complete, and send them.

# Replying to Messages

When you read a message stored in your Mailbox, you can click the Reply or Reply to All link/button. The Compose page appears and allows you to enter the text of your reply message.

If you click the Reply link/button, the original message `Reply-To` header field is automatically placed into the To field of your reply. If the original message did not have the Reply-To header, the original message `From` field is used.

If you click the Reply All link/button, the original message `Reply-To/From` address is copied to the `To` field of the reply message. Then all addresses from the original message `To` fields are added to the reply `To` field, and all addresses from the original message `Cc` fields are copied to the reply `Cc` field.

The text of the original message is formatted as a quotation and copied into the message body text area. The following Settings control the formatting process:

| Message Composer | | |
|---|---|---|
| Reply Header: | default(On ^T^N ^F wrote:) | ^T=time, ^F=sender, ^N=new line |

Reply Header
> This string is added in front of the original message quoted text. The string can contain special symbol combinations which are substituted with the original message data:
> - `^T`   the date and time when the original message was sent
> - `^F`   the From address of the original message
> - `^N`   the EOL (end of line) character(s)

> **Note:** if the Reply Header field is an empty string, the original message text is not automatically added to the reply text.

Each line of the copied original message text is prefixed with this string.

**Note:** to use the standard "flowed" format, use the default "> " Reply Quoting string.

**Note:** If you set this option to an empty string, the original message text will be included into a reply message "as is", and your signature will be added in front of the original message and the Reply Header.

# Forwarding Messages

When you read a message stored in your Mailbox, you can click the Forward link/button. The Compose page appears and allows you to specify the address(es) to which the message should be forwarded and a comment that will be sent along with the body of the forwarded message.

You can view the original message below the message body text area. The unmodified text of the original message is sent along with your comment, using the standard MIME format for message forwarding.

# Attaching Files

You can attach one or several files to a message you want to send. Click the `Browse` button (or a similar button your browser displays for "file-type" fields) and select a file on your local disk (i.e. on the disk attached to the computer that runs your Web browser software). The name of the selected file appears in the Attachment field. Use other Attachment fields to send several files with your message.

**Note:** you should attach files after all other message fields are filled. If you click any button (for example, a button that opens the Address Book), the file selection will be cleared and you will have to select the files again.

The Composer page allows a user to specify the message subject, to enter the recipient address(es), to specify if the DSN (Delivery Status Notification) and/or MDN (Message Disposition Notification) is required, to enter the message text, and to attach files to a message.

# Attaching Your Contact Data

You can compose the `profile.vcf` file containing your personal information using the [Public Info](#) editor.

To attach this Contact (vCard) data file to the composed message, select the My Profile vCard checkbox in the Attachments panel.

# Checking Spelling

If the System Administrator has configured one or several [Spell Checkers](#), the Composer page contains the Check Spelling button with the list of available Spell Checkers (languages).

Select the language and click the Check Spelling button to check the message text. If the selected Spell Checker finds a word it cannot recognize, the Spell Checker panel appears:

| English-GB |
|---|
| Dear John, I won't be able to attdend the meeting on Friday, though we can |
| attend |

1159
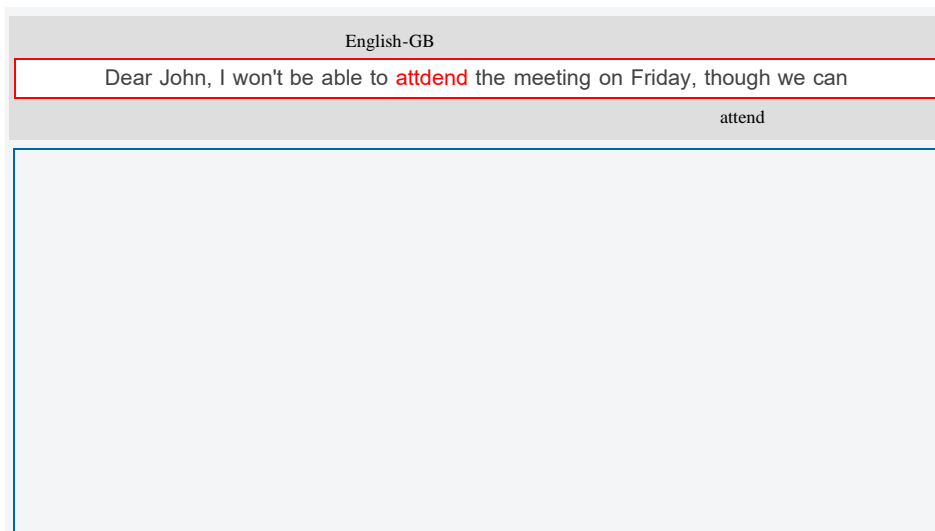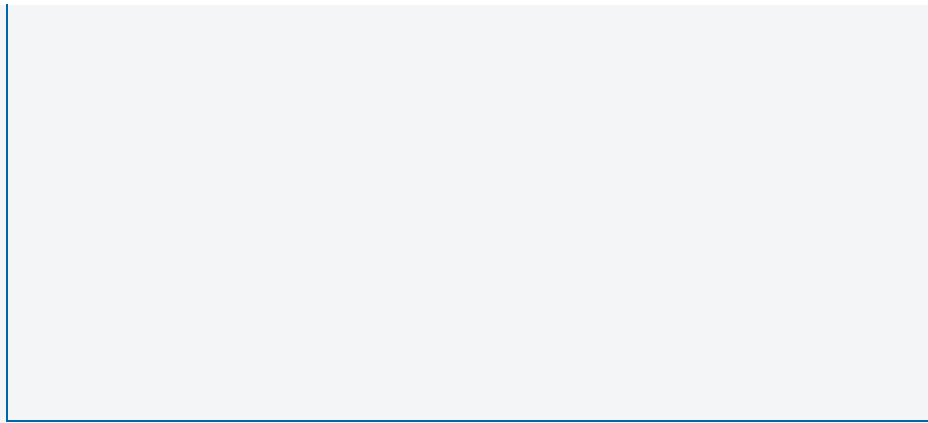
The panel contains a highlighted unrecognized word within surrounding text. Click the Ignore button to continue checking. If the Spell Checker program can suggest changes, the Change to button appears with a menu of the suggestions. Click the Change to button to use the selected suggestion and to continue the spelling check process.

# Delivery Status Notification

You can request a Notification when your message is delivered to the recipients.

Select the Delivery Notification (Notify when Delivered) option to request a DSN. DSN messages sent back to your INBOX tell you that your messages have been successfully delivered to the recipient Mailboxes. They do not tell you if the recipient has actually seen/read your message.

**Note:** The DSN is guaranteed to work only when you send a message to other users on the same CommuniGate Pro Server. If a message is sent to a remote recipient, the remote server that serves that recipient account may or may not support the DSN feature. In many cases your CommuniGate Pro server can detect that a remote server does not support DSN. In this case your server will send you a DSN message itself, telling you that your message has been relayed to a remote host.

# Message Disposition Notification

You can request a Notification when your message is read by the recipient(s) (is displayed to the recipient(s)).

Select the Disposition Notification (Notify when Read) option to request an MDN. MDN messages sent back to your INBOX tell you that your messages have been displayed to the recipients.

**Note:** The MDN is not guaranteed to work. MDN messages are generated with the mail client software, and many mail applications simply do not support MDNs. The client mailers that support MDN may have it disabled, or may ask the user if the MDN message should be sent, and the user may tell the mail application not to send an MDN.

# Address Book

The WebUser Interface allows you to update and use the Address Book. Select an Address Book and click the Display button to open the selected Address Book.

See the Contacts section for more details.

| SharedContacts | **Filter:** |
|---|---|
| John S.Smith <j.smith@company.dom> Thomas Gates <thomas@smithtoyscorp.dom> | |

```
John S.Smith <j.smith@company.dom>
Thomas Gates <thomas@smithtoyscorp.dom>
```

`Close`
Click this button to close the Address Book panel.

`To, Cc, Bcc`
Select one or several addresses in the list, and click one of these buttons to add the selected address(es) to the message you are composing.

`Delete`
Select one or several addresses in the list, and click this button to delete the selected address(es) from the address book.

`Add New`
Type or paste an E-mail address into the field on the right side, and click this button to add the address to the address book.

WebMail Mailboxes Messages Composing Contacts Calendar Tasks Files Notes S/MIME

# WebMail: Contacts

- **Contacts Mailboxes**
  - **Main Contacts Mailbox**
  - **Creating Contacts Mailboxes**
- **Contacts Mailbox Browsing**
- **Calling the Contact**
- **Creating and Editing Contact Items**
- **Creating and Editing Contact Group Items**
- **Address Books**
- **DataSet Address Books**
- **Directory Address Books**
- **Contacts and Address Book Settings**
- **Importing and Exporting Contacts data**

The CommuniGate Pro WebUser Interface allows you to manage your Contacts information.

The standard-based vCard storage format is compatible with any standard-based vCard client, and with Microsoft Windows groupware applications, including Microsoft Outlook (via the MAPI Connector component).

# Contacts Mailboxes

Contact Mailboxes ("folders") can be created in your Account using the WebUser Interface or a MAPI client application (such as Microsoft Outlook). These Mailboxes appear on the Mailboxes page. Click on a Contacts-type Mailbox name to open it.

## Main Contacts Mailbox

You can have several Contacts-type Mailboxes in your Account, but only one of those Mailboxes is assigned the role of the *Main Contacts Mailbox*. When you receive a message with a vCard attachment, the vCard can be stored in the Main Contacts Mailbox.

## Creating Contacts Mailboxes

To create a Contacts-type Mailbox, open the Mailboxes page and select the `Address Book` value in the Create pop-up menu. Type the name of the Contacts Mailbox you want to create, and click the Create button:

| **Management** |
|---|
| |
| Mailbox |

# Contacts Mailbox Browsing

You can browse a Contacts Mailbox by clicking its name (link) on the Mailboxes page.

The Contacts browser page presents the Contacts folder data as a list of Contact Items, with the Item "Formatted Name" and E-mail data displayed:

| Contact |
|---|
| 20        Search:        **8 selected** |
| **\* A B C D E F G H I J K L M N O P Q R S T U V W X Y Z** |
| ✉ ▶ <br> John Smith <br> j.smith@company.dom |
| ✉ ▶ <br> Colt, Susan <br> susan@coltfamily.dom |
| ✉ ▶ <br> Network Group <br> [Network engineers in the South] |
| ✉ <br> 🚩        C1 |
| **Folder Management View As Folder** |

---
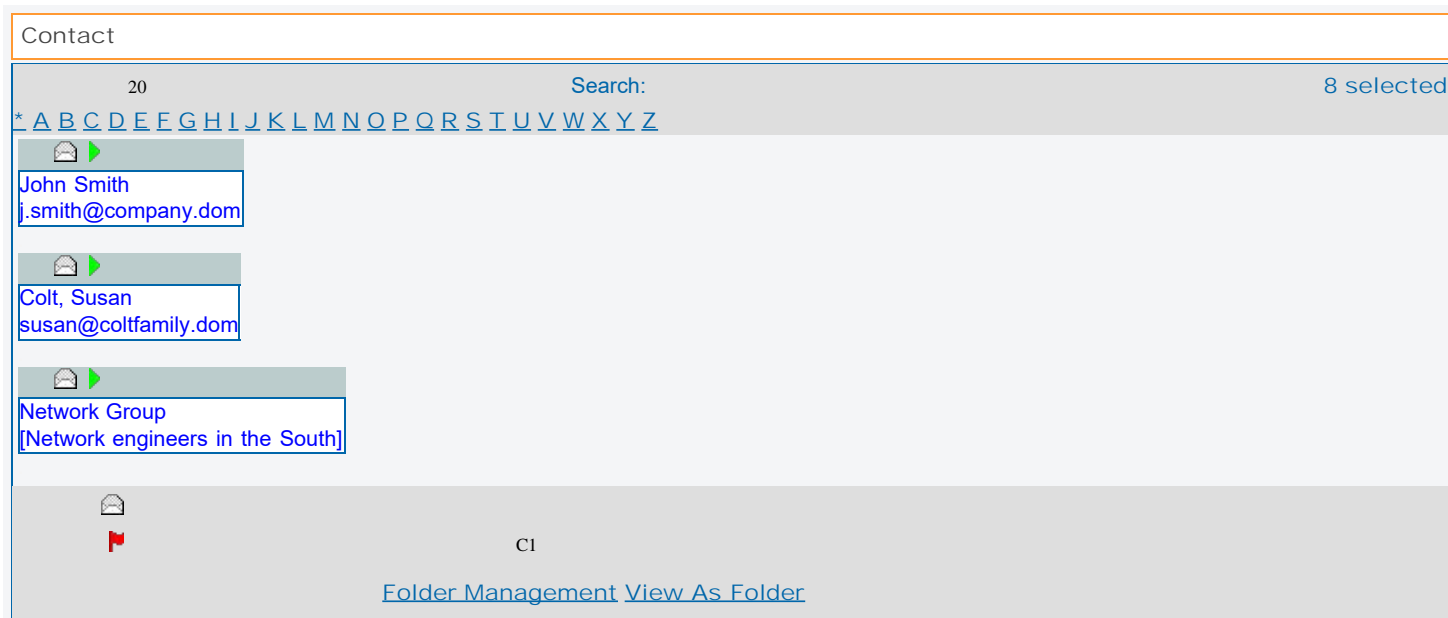
# Calling the Contact

Attribute of the `TEL` type are displayed as links. Click this link to open a separate Make Call window and to initiate a phone call to the specified number.

The CommuniGate Pro Server will call all your currently connected (*registered*) SIP devices. When you accept the call on one of those devices, the Server instructs the device to place a call to the selected phone number, monitors the call progress for some time and disconnects from the device.

| user@domain.dom calling 8002624722 |
|---|
| `<sip:4153837164@domain.dom>` <br> `no device registered` |
| `<sip:8002624722@domain.dom>` <br> `you have answered` <br> `transferring` <br> `Ringing` |

The Make Call window shows the progress log of your calls.

The Server waits for 15 seconds before canceling the call to your own SIP devices.

---

# Creating and Editing Contact Items

Click the New Contact link on the Contacts browser page to create a new Contact Item.

The Contact Composing page contains entry fields for Contact Item elements. Fill all or some fields and click the Save Contact button.

When you need to enter several Contact Items, click the Save and Open New button to save the current Contact Item and to create a new Contact Item without returning to the Contacts browser page.

To modify a Contact Item, open it using its link on the Contact Mailbox browser page, and click the Edit Contact link.

---

# Creating and Editing Contact Group Items

A Contact Group Item contains a list of names and E-mails.

Click the New Contact Group link on the Contacts browser page to create a new Contact Group Item. The Contact Group Editor page contains the list of the Group elements and the Group Note field:



To add an element to the Group, type it in the text field and click the Add New button.

The Contact Group Editor page contains the Address Book panel (see below). You can open an Address Book, select one or several elements, and click the Add to Group button to add those elements to the Group.

Select one or several Group Elements and click the Delete button to remove the selected elements from the Group.

Click the Save Group button to save the Group in the Contacts-type Mailbox.

When you need to enter several Contact Group Items, click the Save and Open New button to save the current Contact Group Item and to create a new Contact Group Item without returning to the Contacts browser page.

To modify a Contact Group Item, open it using its link on the Contact Mailbox browser page, and click the Edit Group link.

# Address Books

The CommuniGate Pro WebUser Interface provides Address Book functionality. Address Books can be used to select E-mail addresses when you compose messages and meeting requests, and when you compose Contact Groups.

There are several sources for E-mail information that can be used as Address Books:

- all Contacts-type Mailboxes.
- Directory subtrees you have selected.
- account DataSets.

The Address Book panel allows you to select the information source, enter the filter text, and display all information source records that match the filter text:

```
John Smith <j.smith@company.dom>
[@] Jim Gray <jim@noc.dom>
[Network engineers in the South]
```

Select the information source from the pop-up menu and click Display to open the Address Book panel. Click the Close button to close the panel.

Select one or several Address Book elements and click the To/Cc/Bcc button to add the selected addresses to the mail message or the meeting request you are composing. Click the Add to Group button to add the selected addresses to the Contact Group Item you are composing.

If an Address Book element is a group, it is displayed in brackets (`[`*name*`]`). When you add a group to the message or the Contacts Group you are composing, all group elements are added.

An Address Book element containing a Certificate has the `[@]` marker. You can send encrypted messages to those recipients.

To add an element to the opened Address Book, type the E-mail address (with an optional *real name* or *comment*) in the text field, and click the Add New button.

To remove elements from the opened Address Book, select them and click the Delete button.

---

# DataSet Address Books

DataSet Address Books are implemented as subsets of the Account DataSet. These Address Books are quite simple, and they can store only the E-mail, Real Name and the Certificate information. These Address Books can be used as String Lists.
Because of the dictionary-based subset design, these Address Books become ineffective if they contain more than 100-500 records.

To specify which subdictionaries of the Account DataSet to show as Address Books, open the Settings page:

| Settings: Contacts | General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info |
|---|---|
| **Dataset Address Books** | |
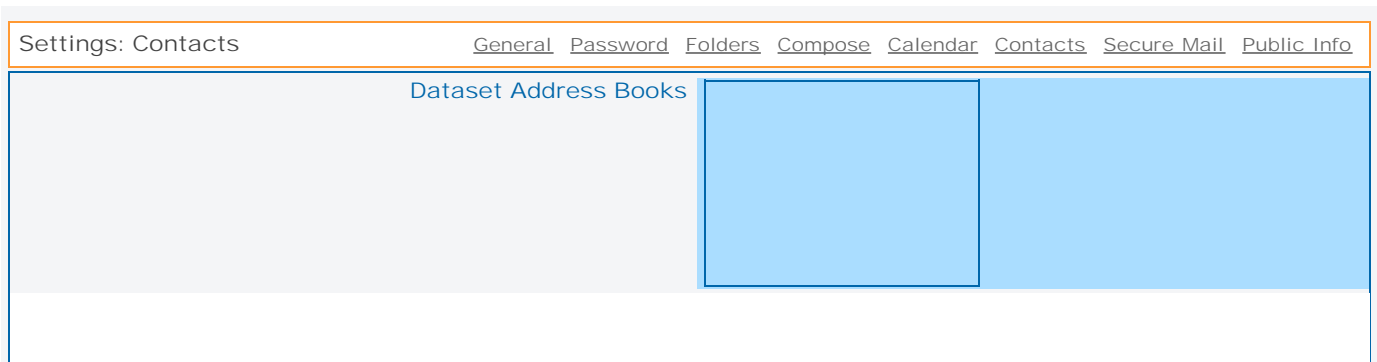
To add an Address Book, enter the DataSet subset name into the last empty field, and click the Update button.

To remove a DataSet Address Book, delete its name and click the Update button. This operations removes the Address Book from the list of the displayed Address Books, it does not delete the Account DataSet subset or its contents.

---

# Directory Address Books

Directory Address Books allow you to search the CommuniGate Pro Directory, including all its Local and Remote Units.

To create a Directory Address Book, open the Settings page:

| Settings: Contacts | General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info |
|---|---|

Enter the Directory Address Book name and the Search Base (*Search DN*) into the last empty fields, and click the Update button.

The special $domain$ Search Base is converted into the DN of the Directory record created for your CommuniGate Pro Domain. You can use the special top Search base to search the entire Directory tree.

To remove a Directory Address Book, delete its name and click the Update button. This operation removes the Address Book from the list of the available Address Books only, it does not remove any Directory data.

# Contacts and Address Book Settings

The Settings pages allow you to specify various Contacts and Address Books options.

You can specify the name of your Main Contacts Mailbox:

| Settings: Contacts | General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info |
|---|---|
| **Main** Main Contacts Mailbox | default(Contacts) |

When you use the File vCard operation, the vCard is stored in your Main Contacts Mailbox.

You can specify the name of your Main Address Book:

| Settings: Contacts | General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info |
|---|---|
| **Main** Main Address Book | Contacts |

The Address Book panel displays the *Current Address Book*. When you login, the Main Address Book is used as the Current Address Book. When you select a different Address Book in that panel, it becomes the Current Address Book.

The Current Address Book is used:

- to store data when you use the Take Address and Take Certificate operations
- to look for recipient Certificates when you are sending an encrypted message

# Importing and Exporting Contacts Data

To import Contacts (vCard) data into a Contacts-type Mailbox, open its Folder Management page. The page contains the Import Contacts Data control:

| no file selected | **Export VCard 2.1 Data**<br>**Export VCard 3.0 Data** |
|---|---|

1166

Use the Browse button to select a text file with vCard data, and click the Import vCard Data button.

If there is an error in text file format, the error message is displayed indicating the text line that caused the problem, and no data is imported (even if some vCard data elements were parsed without errors).

To export contact data in the old (2.1) or new (3.0) vCard format, click one of the Export vCard Data links.

| Introduction |
| Installation |
| SysAdmin |
| Network |
| Objects |
| Storage |
| E-mail |
| Real-Time |
| Access |
| Services |
| Directory |
| Clusters |
| Applications |
| Miscellaneous |
| Licensing |
| WebMail |
| Pronto! |
| PBX |

WebMail **Mailboxes** Messages Composing Contacts **Calendar** Tasks Files Notes S/MIME

# WebMail: Calendar

- **Calendar Mailboxes**
  - **Main Calendar Mailbox**
  - **Creating Calendar Mailboxes**
- **Calendar Browsing**
- **Creating Calendar Events**
  - **Creating Recurrent Events**
- **Replying to Meeting Requests**
- **Reconfirming and Declining accepted Requests**
- **Canceling an Event and Attendee Removing**
- **Processing Event Replies**
- **Calendaring Settings**
- **Importing and Exporting Calendaring data**
- **Server-side Alarms**
- **Automatic Request Processing**

The CommuniGate Pro WebUser Interface allows you to manage your Calendar information (meetings, appointments, events, etc.)

The standard iCalendar format is used to present the Calendaring information, providing compatibility with all standard-based groupware clients, and with Microsoft Windows groupware applications, including the Microsoft Outlook (via the MAPI Connector component).

The Calendar information can be accessed via XIMSS and CalDAV protocols.
Older groupware applications can subscribe to Calendaring data using the HTTP Publish/Subscribe (ICS) method.

The Calendar information can be exported as a text file in the VCALENDAR format.

The WebUser Interface Calendaring functions are available only if the WebCal Service is enabled in the Account settings and in the Account Domain settings.

# Calendar Mailboxes

Calendar Mailboxes ("folders") can be created in your Account using the WebUser Interface or a MAPI client application (such as Microsoft Outlook). These Mailboxes appear on the Mailboxes page. Click on a Calendar-type Mailbox name to open the Calendar.

## Main Calendar Mailbox

You can have several Calendar-type Mailboxes in your Account, but only one of those Mailboxes is assigned the role of the *Main Calendar Mailbox*. When you accept an invitation or create a new appointment or a new meeting request, the calendaring data is stored in the Main Calendar Mailbox.

When the Main Calendar Mailbox is updated, the CommuniGate Pro WebUser Interface removes the current Free/Busy information - it deletes the `freebusy.vfb` file in the File Storage. When someone is trying to access that file, it is rebuilt using the modified information stored in your Main Calendar Mailbox.

## Creating Calendar Mailboxes

To create a Calendar-type Mailbox, open the Mailboxes page and select the `Calendar` value in the Create pop-up menu. Type the name of the Calendar Mailbox you want to create, and click the Create button:

**Management**

Calendar

---

# Calendar Browsing

You can browse a calendar by clicking its name (link) on the Mailboxes page.

The Calendar browser page displays the calendar folder data as a scheduling table:

| Calendar | | | | | | | | | | | | Monthly  Weekly  Daily  Schedule |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **<==** | | **>==<** | | **(-)** | | **(+)** | | | **<==>** | | | **==>** |
| ∧ | 2006 | 8:00AM | 9:00AM | 10:00AM | 11:00AM | | 12:00PM | 1:00PM | 2:00PM | 3:00PM | 4:00PM | 5:00PM | 6:00PM | 7:00PM |
| ‖ | Fri, 01-Dec | | | | Group Meeting | | | Presentation | | | |
| (-) | Sat, 02-Dec | | | | | | | | | | |
| (+) | Sun, 03-Dec | | | | | | | | | | |
| | Mon, 04-Dec | Out of Office | | | Group Meeting | | | Conf Call w/ACME | | | |
| | Tue, 05-Dec | | ACME Meeting | | | | | | | | |
| | Wed, 06-Dec | | | | Group Meeting | | | | | | |
| | Thu, 07-Dec | | | | | | | | | | |
| ‖ | Fri, 08-Dec | | | | | | | | | | |
| V | Sat, 09-Dec | | | | | | | | | | |

**Folder Management**                                    **View As Folder**

The table uses different colors for working and non-working hours and days, and it shows all scheduled events and appointments. Click an event to open it.

Click the arrows in the table corner to move the "visible window" to earlier or later hours, and to previous or next days. Click the `(+)` and `(-)` links to display more or fewer elements.

The `<==>` and `>==<` elements allow to control the time scale used. By specifying larger time slices you can see more time intervals in a smaller window, but then adjusting events can be displayed as conflicting ones.

Click the `View as Folder` link to see the data as a regular Mailbox.

The month Calendar table displays the current month and allows you to switch to a certain day quickly, by clicking the day number:

| 30-Nov-06 6:03:29PM | | | | | | |
|---|---|---|---|---|---|---|
| **<==** | | **Nov, 2006** | | | | **==>** |
| **Mon** | **Tue** | **Wed** | **Thu** | **Fri** | **Sat** | **Sun** |
| | | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | | | |

Use the arrow links to switch to the previous month or to the next month.

| Calendar | | | | | | | | | | | | Monthly  Weekly  Daily  Schedule |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **<==** | | **>==<** | | **(-)** | | **(+)** | | | **<==>** | | | **==>** |
| ∧ | 2006 | 8:00AM | 9:00AM | 10:00AM | 11:00AM | | 12:00PM | 1:00PM | 2:00PM | 3:00PM | 4:00PM | 5:00PM | 6:00PM | 7:00PM |
| ‖ | Fri, 01-Dec | | | | Group Meeting | | | Presentation [X] | Conf Call | | |
| (-) | Sat, 02-Dec | | | | | | | | | | |
| (+) | Sun, 03-Dec | | | | | | | | | | |
| | Mon, 04-Dec | Out of Office | | | Group Meeting | | | Conf Call w/ACME | | | |
| | Tue, 05-Dec | | ACME Meeting | | | | | | | | |
| | Wed, 06-Dec | | | | Group Meeting | | | | | | |
| | Thu, 07-Dec | | | | | | | | | | |
| | Fri, 08-Dec | | | | | | | | | | |

1169

**Folder Management**                                    **View As Folder**

If you have 2 conflicting Events in your Calendaring folder, their table cells are highlighted (as the cell containing the `Conf Call` Event above). The `[x]` link can be used to open the conflicting event.

**Note:** two events can be displayed as conflicting when their time slices do not intersect, but the Calendar view is "too raw" to show them as separate events. For example, if the Event A takes place from 8:00 till 9:00 and the Event B takes place from 9:00 till 10:00, and the Calendar "time slice" is 2 hours, both events end up in the same calendar table cell, so that cell (8:00-10:00) is shown highlighted, with only Event A displayed and the `[x]` link opening the Event B.

---

# Creating Calendar Events

Click the New Event link to create a new Calendaring Event. The page used to compose an Event is a modification of the E-mail composing page.

The Event Composing page contains the controls used to specify the time of the Event:

|  | Starts: | Fri, 01-Dec | 11:00AM | All-Day Event |
|---|---|---|---|---|
| **When:** | Duration: | 60 min | | |
|  |  | | Daily | |

You can specify the time when the event starts, and the duration of the event. To compose an All-Day event, select the All-Day Event checkbox.

The Options panel allows you to specify the Event options:

| **Status:** | Busy |
|---|---|
|  | Private Item |
|  | Send Requests |
| **Priority:** | Normal |

The Event Priority can be High, Normal, or Low.

You can specify how the Event should be marked in your Free-Busy data. The Event time can be marked as `Free`, `Busy`, `Tentative`, or `Out of Office/Unavailable`.

If you select the Private Item option, this Event will be invisible for other users who have access to your Calendar Mailbox.

You can specify the Location and the Subject (Summary) of the Event. The Subject text is used to display the Event on the Calendar view page.

To organize a *meeting*, add the attendees to the `To`, `Optional`, and `Inform` fields:

| **Organizer:** | "Bob" <bob@company.com> |
|---|---|
| **To:** | |
| **Subject:** | |
| **Optional:** | |
| **Inform:** | |
| **Charset:** | Western European (ISO) |
| **Where:** | |

When you save or update a meeting, a meeting request is sent to all attendees. If you want to compose or update a meeting without sending meeting requests, disable the Send Requests option.

The Event composing page displays a list of all specified attendees, along with their confirmation status data. When you receive replies to your meeting request (see below), the attendee confirmation status data is updated. You can also set the status manually if an attendee replied with a non-calendaring E-mail that cannot be processed automatically, or if an attendee replied by other means, such as a phone call.

Click the Show Availability button to display the attendee's free/busy information:

| Attendees | | | | | | | | | | | |
|-----------|-------|-------------------------------------------------------------------------------------------------------|
| **Name** | **State** | 12:30PM  1:00PM  1:30PM  2:00PM  2:30PM  3:00PM  3:30PM  4:00PM  4:30PM  5:00PM |
| joe@company1.com | UNCONFIRMED | |
| b.smith@company2.com | TENTATIVE | |
| susan@company2.com | UNCONFIRMED | |

Click the Save button to store the Event and to send meeting requests to the Event attendees. If you have opened the Event composing page using a link on a Calendar view page, the newly created Event is stored in that Calendar folder (Mailbox). Otherwise the newly created Event is stored in your Main Calendar folder.

You can open an existing Event in your Calendar folder, and click the Edit Event link to update the Event data. When you save the updated Event, invitations are sent to all attendees again (unless you disable the Send Requests option).

## Creating Recurrent Events

You can create a recurrent Event - an Event that repeats at specified dates. Select the *frequency mode* and click the Add Recurrence button (see above). The following recurrent patterns are available:

- Every day or every *N*th day. Use the Daily frequency:

  | | | |
  |---|---|---|
  | Every: | day(s) | |
  | End by: | Never | |

- Every week or every *N*th week, on specified weekdays. Use the Weekly frequency:

  | Every: | week(s) | | | | | | |
  |--------|---------|-----|-----|-----|-----|-----|-----|
  | on: | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
  | End by: | Never | | | | | | |

- Every month or every *N*th month, on the specified day of month. Use the Monthly by Day frequency:

  | | | |
  |---|---|---|
  | Every: | month(s) on day | |
  | End by: | Never | |

- Every month or every *N*th month, on the specified week and weekdays of month. Use the Monthly by WeekDay frequency:

  | Every: | month(s) on | first | | | | | |
  |--------|-------------|-------|-----|-----|-----|-----|-----|
  | | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
  | End by: | Never | | | | | | |

- Every year or every *N*th year, on the specified day of year. Use the Yearly by Day frequency:

  | | | | |
  |---|---|---|---|
  | Every: | year(s) on | of | Jan |
  | End by: | Never | | |

- Every year, on the specified weekdays of specified week. Use the Yearly by WeekDay frequency:

  | | | | |
  |---|---|---|---|
  | Every: | Jan | on | first |

|  | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|---|
| End by: | Never | | | | | | |
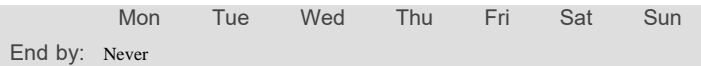
Use the End By control to specify when the Event Recurrence should stop. If you select the Never value, the recurrence will continue infinitely.

Click the Remove Recurrence button to switch back to a one-time Event.

# Replying to Meeting Requests

When you open a meeting request letter, the Event Reply buttons are displayed:



If you click the Accept or Tentative button, a positive reply is sent back to the Event organizer, and the Event is copied into your Main Calendar folder (Mailbox).

If you click the Decline button, a negative reply is sent back to the Event organizer, and the Event is not stored in your Main Calendar folder.

You may want to enter a comment into the panel text field.

If you click any of the Accept/Tentative/Decline buttons, the original request letter is deleted.

# Reconfirming and Declining accepted Requests

When an event is stored in your Calendaring folder, you may want to re-send a confirmation to the event organizer. Open the Calendaring folder and open the Event. The same Event Reply buttons are displayed. Click the Accept or Tentative button to send a positive response to the event organizer.

You may want not to attend an event that you have already acknowledged. Open the Event in your Calendaring folder, and click the Decline button. A negative response is sent to the Event organizer and the Event is removed from your Calendaring folder.

# Canceling an Event and Attendee Removing

If you are the Event organizer, you can cancel the Event by opening the Event in your Calendar, and clicking the Remove Event From Calendar button:



The Cancellation message is sent to all Event attendees and the Event is removed from your Calendaring folder.

You can open an existing Event in your Calendaring folder and remove some of the Event attendees. When you store the Event, the Cancellation message is sent to all removed attendees.

# Processing Event Replies

When you receive a Reply to your Meeting Request message, the Update Attendee Status button appears:

Click this button to change the Attendee status in the Event stored in your Main Calendar folder and to delete this Reply message.

When you receive a Cancellation message for an event you have accepted, the Remove Event From Calendar button appears:

Click this button to remove the Event from your Main Calendar folder.

# Calendaring Settings

The Settings pages allow you to specify the Calendaring Options.

You can specify the name of your Main Calendar Mailbox:

| Settings: Calendar | General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info | |
|---|---|---|
| **Folders** | Main Calendar | Calendar |

You can specify your Work Week parameters: working hours, the first weekday, working weekdays (to specify custom working days, set the pop-up menu to `Custom` and click the Update button):

| Settings: Calendar | General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info | |
|---|---|---|
| **Work Week** | Default | Mon Tue Wed Thu Fri |
| | Starts at | default(Mon) |
| | Working Hours | default(8:00AM)  -  default(5:00PM) |

The Calendar View panel allows you to customize the Calendar browser:

| Settings: Calendar | General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info | |
|---|---|---|
| **Calendar View** | Days to Display | default(7) |
| | Time to Display | default(10 hour(s)) |
| | Time Slice | default(60 min) |
| | Day by Day | default(Yes) |

Your Main Calendar data is used to generate your Free/Busy information. Use the Free/Busy Publishing panel to specify the time period covered by that Free/Busy information:

| Settings: Calendar | General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info | |
|---|---|---|
| **Free/Busy Publishing** | Days to Publish | default(60) |

# Importing and Exporting Calendar Data

To import Calendaring data into a Calendar-type Mailbox, open its Folder Management page. The page contains the Import Calendar Data control:

| | |
|---|---|
| no file selected | **Export Calendar Data** |

Use the Browse button to select a text file with vCalendar or iCalendar data, and click the Import Calendar Data button.

If there is an error in text file format, the error message is displayed indicating the text line that caused the problem, and no data is imported (even if some calendar data elements were parsed without errors).

All iCalendar import file items that have the CANCEL method are used to remove existing items from the Calendar-type Mailbox. All other items are "published" in that Mailbox, i.e. they are stored in the Mailbox and all other items with the same UID are removed from the Mailbox.

To export calendaring data in the iCalendar format, click the Export Calendar Data link.

# Server-side Alarms

You may want to get Event Alarms even when you are not logged into the WebUser Interface, or any other Calendaring client. The CommuniGate Pro server can send you Event Alarms as E-mails, Instant Messages, or Phone calls.

To enable Server-side Alarm processing, open the Calendar Settings page and use the Server-side Processing panel:

| Settings: Calendar | General Password Folders Compose Calendar Contacts Secure Mail Public Info | |
|---|---|---|
| **Server-side Processing** | Send Alarms as IM | default(Yes) |
| | Send Alarms as Phone Call | No |
| | Send Alarms as E-mail | default(Yes) |

If an Event in the Calendar has an enabled Alarm element (usually specified as time before the Event start), then you will receive an Instant Message, E-mail, or a phone call notifying you about the upcoming Event.

# Automatic Request Processing

You can set your Account to be scheduler for some resource, such as a conference room, company car, etc.

When an Account LargeConfRoom@mycompany.com is created, log into that account and make sure it has the Main Calendar Mailbox created (you may want to create a test Event, and when it is being processed, the Main Calendar is created and the Event is stored there).

Then open the Rules page and create the following `Process Requests` Rule:

| Data | Operation | Parameter |
|---|---|---|
| Source | is | |
| **Action** | **Parameter** | |
| Accept Request | | |
| | | |
| Reject with | | |

The Accept Request Rule action parses the incoming message and tries to find a Calendar Event Request or Calendar Event Cancel object in the message. If the message does not contain any Calendar Event parts, this Rule action does nothing and the next action rejects the message sending an error report to the message sender. If the rule parameter contains the `[tasks]` prefix, then the server looks for Task Request or Task Reply objects in the message.

If the Event or Task Request is found, the Accept Request Rule action opens the Calendar or Tasks Mailbox specified as the Rule parameter or the Main Calendar or Main Tasks Mailbox if the Rule parameter is an empty string (after processing possible prefixes).
The Rule then checks that the requested event does not conflict with any other object. If a conflicting object is found, a negative Reply is sent to the organizer. The negative Event Reply contains the conflicting Event organizer's name. The incoming message is discarded, and Rule processing stops.

If no conflicting Event is found in the Calendar, the message is copied into the Main Calendar and a positive Event Reply is sent to the Event organizer. The Account [Free/Busy information](Free/Busy information) is updated. The incoming message is discarded, and Rule processing stops.

You can specify the `[ignore-conflicts]` prefix in the Rule action parameter field. In this case the Accept Request Rule action will not check for conflicting events or tasks.

The prefix `[tentative]` can be also added to the parameter field so the event or task request are accepted tentatively.

You may want to grant certain persons a right to acquire a resource time slice even if it has been booked by someone else. You need to create a different Rule (for example, with the "Process VIP Requests" name) and assign a higher priority to that Rule:

| Data | Operation | Parameter |
|---|---|---|
| Source | is | |
| Return-Path | in | |
| **Action** | **Parameter** | |
| Accept Request | | |
| Reject with | | |

If a message is processed with the Accept Request Rule action with the `[force]` parameter, and a conflicting Event is found in the Calendar:

- if the conflicting Event is not recurrent, or the new Event is recurrent, the conflicting Event is removed and a negative Event Reply is sent to the conflicting Event organizer.
- if the conflicting Event is recurrent, and new Event is not recurrent, the exception date is added to the conflicting Event.

The Accept Reply Rule action parses the incoming message and tries to find a Calendar Event Reply object in the message. If the message does not contain any Calendar Event parts, this Rule action does nothing and the next action can reject the message sending an error report to the message sender.

If the Event Reply is found, the Accept Reply Rule action opens the Calendar Mailbox specified as the Rule parameter or the Main Calendar Mailbox if the Rule parameter is an empty string.
The Rule then tries to process the Event Reply. If processing succeeds, the incoming message is discarded, and Rule processing stops.

CommuniGate Pro

| Introduction |
| Installation |
| SysAdmin |
| Network |
| Objects |
| Storage |
| E-mail |
| Real-Time |
| Access |
| Services |
| Directory |
| Clusters |
| Applications |
| Miscellaneous |
| Licensing |
| **WebMail** |
| Pronto! |
| PBX |

WebMail | Mailboxes | Messages | Composing | Contacts | Calendar | **Tasks** | Files | Notes | S/MIME

# WebMail: Tasks

- **Task Mailboxes**
  - **Main Task Mailbox**
  - **Creating Task Mailboxes**
- **Task List Browsing**
- **Creating Tasks**
- **Assigning Tasks**
- **Replying to Task Assignment Requests**
- **Updating Task Status**
- **Canceling Tasks**
- **Processing Task Assignment Replies**
- **Tasks Settings**
- **Importing and Exporting Tasks data**

The CommuniGate Pro WebUser Interface allows you to manage your To-Do ("Tasks") information.

The standard iCalendar format is used to present the Tasks information, providing compatibility with all standard-based groupware clients, and with Microsoft Windows groupware applications, including Microsoft Outlook (via the MAPI Connector component).

The Tasks/ToDo information can be accessed via XIMSS and CalDAV protocols.
Older groupware applications can subscribe to Tasks/ToDo data using the HTTP Publish/Subscribe (ICS) method.

The Tasks/ToDo information can be exported as a text file in the VCALENDAR format.

The WebUser Interface Tasks functions are available only if the WebCal Service is enabled in the Account settings and in the Account Domain settings.

# Tasks Mailboxes

Tasks Mailboxes ("folders") can be created in your Account using the WebUser Interface or a MAPI client application (such as Microsoft Outlook). These Mailboxes appear on the Mailboxes page. Click on a Tasks-type Mailbox name to open the Tasks list.

## Main Tasks Mailbox

You can have several Tasks-type Mailboxes in your Account, but only one of those Mailboxes is assigned the role of the *Main Tasks Mailbox*. When you accept a task assignment request or create a new Task, the task data is stored in the Main Tasks Mailbox.

## Creating Tasks Mailboxes

To create a Tasks-type Mailbox, open the Mailboxes page and select the `Tasks` value in the Create pop-up menu. Type the name of the Tasks Mailbox you want to create, and click the Create button:

| **Management** |
|---|
| |
| Task |
| |

# Task List Browsing

You can browse a Tasks Mailbox by clicking its name (link) on the [Mailboxes](#) page.

The Tasks browser page displays the Tasks folder data as a scheduling table:

| Tasks | | | | | | | | | Hide Completed |
|---|---|---|---|---|---|---|---|---|---|
| <== | | | (-) | | (+) | | | | ==> |
| 2006 | Sat, 02-Dec | Sun, 03-Dec | Mon, 04-Dec | Tue, 05-Dec | Wed, 06-Dec | Thu, 07-Dec | Fri, 08-Dec | Sat, 09-Dec | Sun, 10-Dec |
| 1 | | | | Urgent: Prepare slides for AC.. (10%) | | | | | |
| 2 | | Create the conference report (0%) | | | | | | | |
| 3 | | | Read the ACME product docs (100%) | | | | | | |

**Folder Management**                    **View As Folder**

The table uses different colors for high-priority, regular, and completed Tasks. Click a Task to open it.

Click the arrows in the table corner to move the "visible window" to earlier or later days, and to page the task list. Click the `(+)` and `(-)` links to display more or fewer elements.

Click the `View as Folder` link to see the folder data as a regular Mailbox.

# Creating Tasks

Click the New Task link to create a new Task item. The page used to compose a Task is a modification of the E-mail composing page.

The Task Composing page contains the controls used to specify the start and Due dates of the Task, the Task completion status and its priority:

| **When:** | Starts: | Tue 01-Jul-03 | 09:30 |
|---|---|---|---|
| | Due: | Tue 08-Jul-03 | 14:00 |

You can specify the time when the Task starts, and when it is due.

The Options panel allows you to specify the Task options:

| **Priority:** | Normal |
|---|---|
| **Complete:** | 20          % |
| | Private Item |
| | Send Requests |

The Task Priority can be High, Normal, or Low.

You can specify the completion status of the Task. If you set the Complete setting to 100%, the Task is marked as *completed*.

If you select the Private Item option, this Task will be invisible for other users who have access to your Tasks Mailbox.

You can specify the Task Subject (Summary). The Subject text is used to display the Task on the Task list page.

# Assigning Tasks

To assign a task to someone else, add an Email address to the `To` field:

| **To:** |
|---|
| **Subject:** |

When you save or update a Task, a Task Assignment request is sent to the specified address. If you want to compose or update a Task without sending a Task Assignment request, disable the Send Requests option.

The Task composing page displays a list of those to whom the Task is assigned, along with their confirmation status data. When your receive replies to your Task Assignment request (see below), the confirmation status data is updated. You can also set the status manually if the person has replied with a non-calendaring E-mail that cannot be processed automatically, or if a reply has come by other means, such as a phone call.

Click the Save button to store the Task and, optionally, to send a Task Assignment request. If you have opened the Task composing page using a link on a Tasks view page, the newly created Task is stored in that Tasks folder (Mailbox). Otherwise the newly created Task is stored in your Main Tasks folder.

You can open an existing Task item in your Tasks folder, and click the Edit Task link to update the Task data. When you save the updated Task, a Task Assignment request is sent again (unless you disable the Send Requests option).

# Replying to Task Assignment Requests

When you open a Task Assignment request letter, the Assignment Reply buttons are displayed:



If you click the Accept or Tentative button, a positive reply is sent back to the Task organizer, and the Task is copied into your Main Tasks folder (Mailbox).

If you click the Decline button, a negative reply is sent back to the Tasks organizer, and the Task is not stored in your Main Tasks folder.

You may want to enter a comment into the panel text field.

If you click any of the Accept/Tentative/Decline buttons, the original request letter is deleted.

# Updating Task Status

When an assigned Task is stored in your Tasks folder, you may want to re-send a confirmation to the Task organizer. Open the Tasks folder and open the Task. The Assignment Reply buttons are displayed.
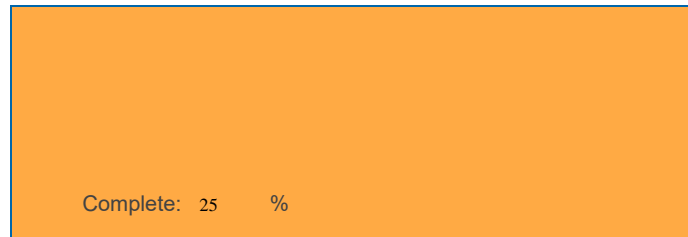


Click the Accept or Tentative button to send a positive response to the Task organizer.

To update the Task completion status, set the new Complete value and click the Update Status button. Your copy of the Assigned Task is updated and an "in-process" or "completed" response is sent to the Task organizer.

You may choose not to perform an Assigned Task that you have already acknowledged. Open the Task in your Tasks folder, and click the Decline button. A negative response is sent to the Task organizer and the Task is removed from your Tasks folder.

# Canceling Tasks

If you are the Assigned Task organizer, you can cancel the Task by opening it and clicking the Cancel Task button:

A Cancellation message is sent to those whom you have assigned the Task to, the Task is removed from your Tasks folder.

You can open an existing Task and remove some assignees. When you store the Task, a Cancellation message is sent to all removed assignees.

## Processing Task Assignment Replies

When you receive a Reply to your Task Assignment request, the Update Assignee Status button appears:



Click this button to change the Attendee status in the Task stored in your Main Tasks folder and to delete this Reply message.

When you receive a Cancellation message for a Task you have accepted, the Cancel Task button appears:
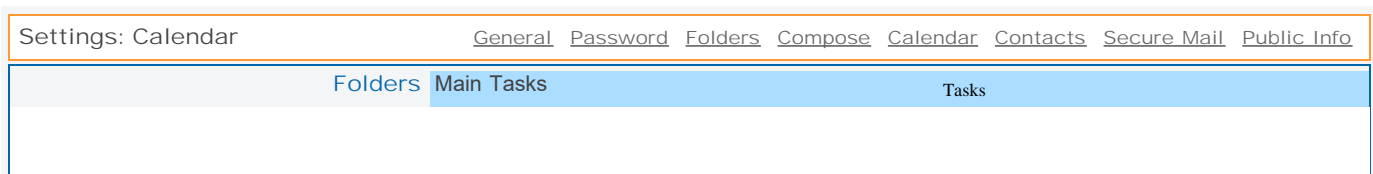


Click this button to remove the Task from your Main Tasks folder.

## Tasks Settings

The Settings pages allow you to specify the Tasks Options.

You can specify the name of your Main Tasks Mailbox:

| Settings: Calendar | General Password Folders Compose Calendar Contacts Secure Mail Public Info | |
|---|---|---|
| **Folders** Main Tasks | | Tasks |
| | | |

The Tasks View panel allows you to customize the Tasks browser:

| Settings: Calendar | General Password Folders Compose Calendar Contacts Secure Mail Public Info | |
|---|---|---|
| **Task View** Days to Display | | default(7) |
| Tasks to Display | | default(20) |
| Show Completed | | default(YES) |
| | | |
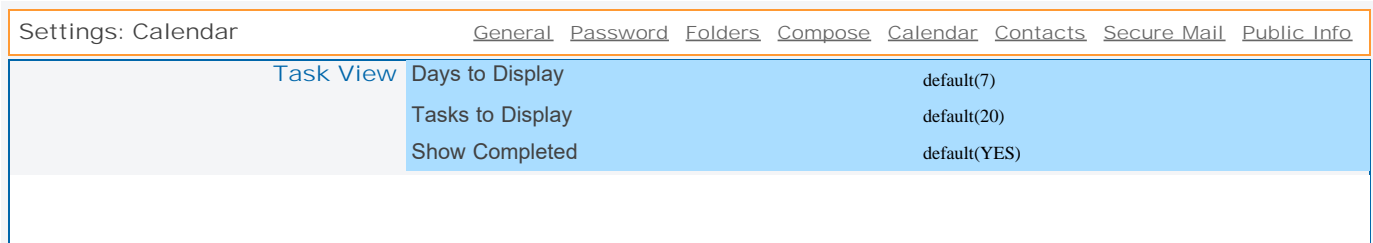
## Importing and Exporting Tasks Data

To import Tasks data into a Tasks-type Mailbox, open its Folder Management page. The page contains the Import Tasks Data control:
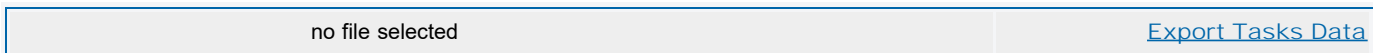
| no file selected | **Export Tasks Data** |
|---|---|

Use the Browse button to select a text file with vCalendar or iCalendar data, and click the Import Tasks Data button.

If there is an error in text file format, the error message is displayed indicating the text line that caused the problem, and no data is imported (even if some iCalendar data elements were parsed without errors).

To export Tasks data in the iCalendar format, click the Export Tasks Data link.

# CommuniGate Pro

| Introduction |
| Installation |
| SysAdmin |
| Network |
| Objects |
| Storage |
| E-mail |
| Real-Time |
| Access |
| Services |
| Directory |
| Clusters |
| Applications |
| Miscellaneous |
| Licensing |
| WebMail |
| Pronto! |
| PBX |

**WebMail** **Mailboxes** **Messages** **Composing** **Contacts** **Calendar** **Tasks** **Files** **Notes** **S/MIME**

## WebMail: Files

- **File Storage Browser**
- **File Access Rights**

The CommuniGate Pro WebUser Interface allows you to manage your Account File Storage.

The File Storage can be used to store files which can be accessed over the Internet, using various clients such as browsers and/or FTP clients. The File Storage and/or its directories can be "mounted" on your desktop computers or mobile devices using the "virtual disk" functionality.

# File Storage Browser

Click the Files link to open the File Browser page. This page displays all the files and file directories on the "top level" of your File Storage:

**Files**

| | Name | Size | Modified | |
|---|---|---|---|---|
| | 2009 | ==> | | Settings |
| | Aug99 | ==> | | Settings |
| | Bodrum1.JPG | 107295 | 02 Mar, 09 | Settings |
| | Bodrum2.JPG | 207889 | 02 Mar, 09 | Settings |
| | private | ==> | | Settings |
| | profile.vcf | 135 | 03-Nov | Settings |
| | pubcal | ==> | | Settings |
| **This Folder:** | 7 | 315319 | | |
| **Totals:** | 560 | 522M | | |
| **Limits:** | 1000 | 1024M | | |
| | no file selected | | | |

Click the Browse button and select a file you want to upload to the File Storage. Click the Upload File button to upload the file. Its name should appear in the list.

Select the checkboxes to mark the files and/or folders you want to remove from the File Storage and click the Delete Marked button. The selected files will be removed.

Type in a name and click the Create Folder button to create a folder (sub-directory) in the File Storage.

Select exactly one checkbox to mark the file or folder you want to rename, and enter a new name for it in the field next to the Rename Marked button. Click the Rename Marked button to rename the selected file or folder.

Click the file name link to open the file. Click the folder name link to open the subdirectory. When a subdirectory is opened, its name is displayed on the top of the file list. Click the UP link to open the parent subdirectory.

The This Folder line displays the total number of files and folders, and the total size of all files in the opened folder. The Totals line displays the total number of files and folders, and the total size of all files in the File Storage. The limits line displays the specified maximum number of files and folders and the specified maximum total file size for this File Storage.

To manage the file Attributes, click the `Settings` link.

# File Access Rights

The File Management page allows you to set the ACL (Access Control List) settings for the selected file or file directory.

| San Diego Photos: File Management | | | | | |
|---|---|---|---|---|---|
| **Access Control List** | | | | | |
| **Identifier** | **List** | **Delete** | **Read** | **Write** | **Admin** |
| | | | | | |
| | | | | | |
| | | | | | |

To grant file access rights to a user, enter the user name into the Identifier field, select the desired access rights, and click the Update button. To grant an access right to everybody, use the word `anyone`. To remove certain rights from a particular user, "grant" those rights to the identifier `-username`. See the File Storage section for more details.

| Introduction |
| Installation |
| SysAdmin |
| Network |
| Objects |
| Storage |
| E-mail |
| Real-Time |
| Access |
| Services |
| Directory |
| Clusters |
| Applications |
| Miscellaneous |
| Licensing |
| **WebMail** |
| Pronto! |
| PBX |

**WebMail** **Mailboxes** **Messages** **Composing** **Contacts** **Calendar** **Tasks** **Files** **Notes** **S/MIME**

# WebMail: Notes

- Notes Mailboxes
    - Main Notes Mailbox
    - Creating Notes Mailboxes
- Notes List Browsing
- Creating And Editing Notes
- Notes Settings

The CommuniGate Pro WebUser Interface allows you to manage your Notes.

Notes are texts with subjects and, optionally, attached files. The standard RFC822 ("mail message") format is used to present the Notes information, providing compatibility with all standard-based groupware clients, and with Microsoft Windows groupware applications including Microsoft Outlook (via the MAPI Connector component).

# Notes Mailboxes

Notes Mailboxes ("folders") can be created in your Account using the WebUser Interface or a MAPI client application (such as Microsoft Outlook). These Mailboxes appear on the Mailboxes page. Click on a Notes-type Mailbox name to open the Notes list.

## Main Notes Mailbox

You can have several Notes-type Mailboxes in your Account, but only one of those Mailboxes is assigned the role of the *Main Notes Mailbox*. When you create a new Note and do not specify explicitly where to store it, the Note is stored in the Main Notes Mailbox.

## Creating Notes Mailboxes

To create a Notes-type Mailbox, open the Mailboxes page and select the `Notes` value in the Create pop-up menu. Type the name of the Notes Mailbox you want to create, and click the Create button:

**Management**

Notes

# Notes List Browsing

You can browse a Notes Mailbox by clicking its name (link) on the Mailboxes page.

The Notes folder is displayed in the same format as a regular mail Mailbox, but only the Subject and Date columns are displayed.

# Creating and Editing Notes

Click the New Note link to create a new Note item. The page used to compose a Note is a modification of the E-mail composing page.

The Note Composing page contains only the Subject header field.

To edit an already created Note, open it and click the Edit Note link.

# Notes Settings

The Settings pages allow you to specify the Notes Options.

You can specify the name of your Main Notes Mailbox:

| Settings: Compose | General Password Folders Compose Calendar Contacts Secure Mail Public Info |  |
|---|---|---|
| **Folders** | Notes Mailbox | Notes |

CommuniGate Pro

# WebMail: Secure Mail (S/MIME)

- **Public Key Infrastructure (PKI)**
- **Digital Signatures**
- **Certificates**
- **Private Key and Certificate Storage**
- **Private Key Activation**
- **Receiving Signed Messages**
- **Recording Certificates**
- **Sending Signed Messages**
- **Sending Encrypted Messages**
- **Receiving Encrypted Messages**
- **Encrypting Stored Messages**
- **Encrypting Incoming Messages**

The CommuniGate Pro Secure Mail (S/MIME) functionality is based on the Public Key technology. Using S/MIME, you can:

- digitally sign your message, so the recipient can:
  - verify that the message has been really sent by you;
  - verify that the message content has not been altered and that it was received in exactly the same form as it was composed by you (the sender);
- digitally encrypt your message, so only the recipients can read it, even if the message has been intercepted while it was being transferred, or if it has been copied from the server files that store the message.

# Public Key Infrastructure (PKI)

The Public Key technology implements a so-called asymmetric cryptography. Using a regular, symmetric cryptography, both parties need to know some "key" or "password" (called "shared secret"). Before the parties can exchange data securely, they need to exchange that "shared secret", and this is the main security problem with symmetric cryptography: the "shared secret" can be stolen during the exchange process.

Imagine a spy who needs to exchange information with his center securely, using some secret key. That key must change frequently to ensure that the time needed to "break the key" is much larger than the "lifespan" of the information encrypted with that key. The center has to send those new keys to the spy (or vice versa), but those keys can be intercepted, and anyone who succeeds in intercepting the key will be able to decrypt all messages they send.

The Public Key technology uses pairs of specially generated keys. Both keys are very large numbers: they have 512 bits in length (approximately 60 decimal digits) or more. The special method used to generate those key pairs and the method used to encrypt information with those keys ensures that the message encrypted with one key can be decrypted with the other key. One key is called the "Private Key", the other key is called the "Public Key".

The PKI algorithms ensure that any data encrypted with the Public Key can be decrypted with the Private Key, any data encrypted with the Private key can be decrypted with the Public Key, and that it is extremely difficult to calculate the Private Key if the Public Key is known. Please note that messages encrypted with the Public Key cannot be decrypted with the same Public Key - they can be decrypted only with the Private Key.

Now we can see how this technology can be used by a spy, or any other party that needs to exchange information securely:

- The spy generates a pair of Public/Private Key using the key generation algorithm.
- The Private Key is stored securely at the spy's location.
- The Public Key is sent to the center using any type of communication - even a very "open" one. For example, the Public Key can be posted on the Web.
- When the center receives the Public Key, it uses it to encrypt the information it wants to send to the spy. It then sends it to the spy using any type of communication - again, it can just post the encrypted message on the Web.
- Since only the spy posses the Private Key, only the spy can decrypt the information the center has sent. All other parties only have the Public Key and the encrypted data, but the Public Key cannot be used to decrypt the information, and it cannot be used to calculate the

Private Key.

- The Center can also generate a pair of Private/Public Keys and send its Public Key to all its spies - so all of them can send message to the Center that only Center can decrypt, using its Private Key.

In real applications, PKI is not used to encrypt actual information. Instead, a random "regular key" ("shared secret","password") is generated, actual information is encrypted using that shared secret, and PKI is used to encrypt that "shared secret" key. The encrypted "shared secret" key is appended to the actual information. The recipient uses its Private Key to decrypt the "shared secret", and then uses that "shared secret" to decrypt the actual data, using regular, "symmetric cryptography".

This method is used to decrease the amount of PKI computations (shared key is usually much smaller than the actual information), since PKI algorithm are much more complex than symmetric-key algorithms.

When it is said that the information is encrypted using 40 bit, or 56 bit, or 128 bit keys, it means that the random "shared secret" key used had this length - the PKI keys have much higher length. It is much easier to break a 40-bit "shared secret" key used to encrypt data, than to break a PKI key used to encrypt that "shared secret". But "shared secret" keys are generated at random for each transaction, so if someone breaks the "shared secret key" used for any transaction, only that transaction will be compromised (decrypted), because other transactions with the same PKI keys will use different "shared secret" keys.

The method with 2 keys (PKI and "shared secret") allows a sender to send an encrypted message to several recipients at once. A message is encrypted using a random "shared secret" key, and then PKI is used to encrypt that "shared secret" several times, with Public Keys of all recipients. All encrypted "shared keys" are appended to the message, and each recipient can find the "shared secret" encrypted with its own Public Key, decrypt it with its Private Key, and use the decrypted "shared secret" to decrypt the actual information.

# Digital Signatures

Encryption alone does not solve all security problems. If we return to our spy/center example, anyone who got the spy's Public Key can send an encrypted message to the spy. The spy needs to verify that the sender is really the center. The "digesting algorithms" and the Public Key algorithms are used to implement digital signatures.

Digest is a relatively short (16-40 bytes) number with a "checksum" of the message. The algorithms used for "digesting" ensure that it is very difficult to compose 2 different messages that would have the same digest values.

To sign a message, the sending software:

- calculates a "digest" for the message;
- encrypts the calculated digest using its own Private Key;
- appends the encrypted digest to the message.

The receiving party uses the sender's Public key (it is known to the receiving party) to decrypt the message digest, calculate the message digest itself, and compare the decrypted and calculated digests. If they match, the message has not been altered, and it was really sent by the party that has the proper Private Key.

In our spy example, a third party won't be able to send a message pretending to be the Center, because it does not know the Center Private Key. And if the third party encrypts the digest with some other Private Key, the signature verification will fail, because the spy will try to decrypt the digest with the Center's Public Key, and the resulting garbage will not match the calculated message digest.

# Certificates

The encryption and signing methods assume that parties can freely exchange the Public Key information. The PKI eliminates the risk of "key stealing": the Public Key can be known to anybody (can be "publicly known").
But there is another risk - when a party receives the Public Key, it should verify that that it really belongs to the proper entity. Otherwise a third party can generate its own Private/Public Key and send the Public Key to the center, pretending that it is the Public Key of the spy. If the center does not detect that this is a fake, it will use that key to encrypt the information it sends to the spy. The spy will not be able to read it (it is encrypted with the wrong Public Key), but the third party that has issued the key pair will be able to decrypt it, as it posses the matching Private Key.

To solve this problem, the Public Keys are not distributed in the "raw form". Instead, they are distributed embedded into *Certificates*. A Certificate contains the following data:

- "Subject" - the name of the party the Certificate belongs to.
- Public Key - the Public Key of the "Subject".
- "Issuer" - the name of the party that has issued this Certificate.
- Signature - the digest of the data above, encrypted with the Issuer's Private Key.

Certificates are issued by a Certificate Authority - some party that all parties choose to trust. All parties should know the Public Key of the Certificate Authority. Modern Internet applications (browsers, mailers, etc.) have a built-in list of *Trusted Authorities* (including VeriSign and

other similar companies), and have the Public Keys of those Trusted Authorities built-in.

When a certificate is received, the receiving party can verify if it has been issued by a "trusted authority": it checks if the "issuer" name in the Certificate is one of the "Trusted Authorities", and uses the already known Public Key of that Authority to verify the Certificate signature. If the signature is verified, the party can trust that the Public Key in the Certificate really belongs to the party specified in the Certificate Subject.

Very often an intermediate Certificate Authority is used. For example, a corporation can get a Certificate issued by a Trusted Authority, and then it can act as a Certificate Authority itself, issuing certificates for its employees. To enable verification of such a certificate by any third party, the Certificates issued by an Intermediate Certificate Authority are sent together with the Intermediate Certificate Authority own Certificate. The receiving party first checks that the Certificate is really issued by that intermediate Authority (by using the Public Key from its Certificate to verify the signature in the sender Certificate), and then it checks that the intermediate Authority is what it claims to be (by verifying its Certificate using the known Trusted Authority Public keys).

# Private Key and Certificate Storage

In order to use PKI for Secure Mail, an Account should have its own Private Key and a Certificate with its Public Key. The Private Key should be protected as much as possible, while the Certificate should be easily accessible by anyone.

CommuniGate Pro stores the Certificate in the Account Settings (as the "userCertificate" element), and also it copies the Certificate into the Directory - if the Directory Integration is enabled.

CommuniGate Pro stores the Private Key in the Account Settings, but it encrypts the Private Key with a "Secure Mail Password". To use any of the Secure Mail functions, you should enter the "Secure Mail Password" to let the server read and decrypt your Private Key.

**Note:** The server does **not** store your Secure Mail Password anywhere. If you forget the password, you will need to obtain a new Private Key and Certificate. This means that your will not be able to decrypt any message encrypted with your old Public Key.
Neither your System Administrator nor CommuniGate Systems will be able to help you get those messages back.

**Note:** While it is very important to remember your Secure Mail Password, it is not too difficult to do: the Secure Mail Password can be a word or a phrase (up to 100 symbols), in any language.

You can use your regular E-mail client (such as Microsoft® Outlook or Netscape® Messenger) to obtain a personal Private Key and Certificate (also called "Digital ID"). You can then export that "Digital ID" to a `.pfx` or `.p12` file - a so-called PKCS#12-formatted file. In order to protect your data, the E-mail client will ask you for a password, and will encrypt the exported information with that password.
**Note:** while the file format supports non-ASCII symbols in a file password, you should use ASCII symbols only, as many E-mail clients (including Outlook) do not process national symbols correctly.

Connect to the Server using the WebUser Interface, and open the Settings section. Click the Secure Mail link to open the page that contains the following fields:

| Settings: Secure Mail | General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info | |
|---|---|---|
| | New Password | |
| | Reenter Password | |
| | | |
| Import Key and Certificate | PFX File | no file selected |
| | File Password | |
| | | |

**Note:** If you do not see the Secure Mail link on your Settings pages, it means that your Account or Domain has the S/MIME service disabled.

Enter the name of the saved `.pfx` or `.p12` file or use the Browse button to select the file on your workstation disks. Enter the File Password you used when you created that file.

Enter the password that will become your Secure Mail Password - this password will protect your Private Key on the CommuniGate Pro server. Enter this password twice, into two fields, and click the Import File Data button. If you have entered the correct File Password, the Certificate and Private Key information will be stored in your CommuniGate Pro Account settings.

Alternatively, you can ask the CommuniGate Pro server to generate a Private Key and a Certificate for you. Use the Generate Key And

Certificate button:

As when importing Key and Certificate from a file, you need to specify the password (twice) that will will become your Secure Mail Password.

The generated Certificate will be issued for the E-mail address you have specified as your `From` address in the WebUser Interface Settings, but only if that address points to your CommuniGate Pro account. Otherwise, the Certificate is issued for your CommuniGate Pro Account address.

The generated Certificate is signed with the CommuniGate Pro server certificate.

The Secure Mail page now shows your Certificate data and the size of the Private Key.

**Settings: Secure Mail**    General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info

**Modify Secure Mail Password**  New Password

Reenter Password

Export Key and Certificate

**Remove Key and Certificate**  PFX File                     no file selected

File Password

To change your Secure Mail Password, enter the new password twice into the Modify Secure Mail Password panel fields and click the Modify Password button.

To store your Key and Certificate information in a file on your workstation disks, click the Export Key and Certificate link. A panel will open in a new window:

**Export Key and Certificate**    General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info

**Export**  File Password

Verify File Password

Enter the password to be used to encrypt your Key and Certificate information in the file (you need to enter it twice), and click the Export button. Your browser should ask you where to save the `CertAndKey.pfx` file (you can rename it).

IF you decide to remove your Private Key and Certificate, you need to have their copy in a file. This is done to ensure that you can restore this info if you removed the Key by mistake. Remember that if you remove the Private Key completely and do not have a file to restore it from, all encrypted messages sent to you will become completely unreadable.
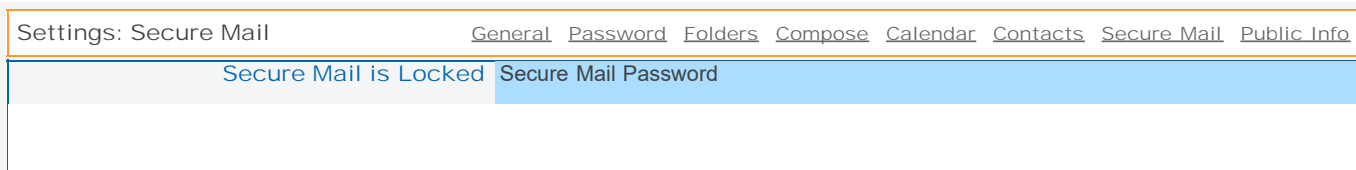
To remove the Key and Certificate, enter the name of the file that has the your Key and Certificate and the file password, and click the

Compare with File and Delete button. CommuniGate Pro will decrypt the file using the supplied password and it will compare it to your current Private Key. If the Keys match, the Private Key and Certificate are removed from your Account Settings.

# Private Key Activation

When the Private Key is placed into the Account Settings, it is *activated*. The WebUser Interface automatically decrypts all messages encrypted with your Certificate/Public Key, and you can send encrypted and signed messages. In order to protect your sensitive information, your Private Key is automatically deactivated ("Locked") every 3 minutes. If you log out of the WebUser Interface session, and then log in again, your Private Key will not be automatically activated.

To activate your Private Key again, you need to enter the Secure Mail Password on any of the CommuniGate Pro WebUser Interface pages that displays the S/MIME Key Activation panel:
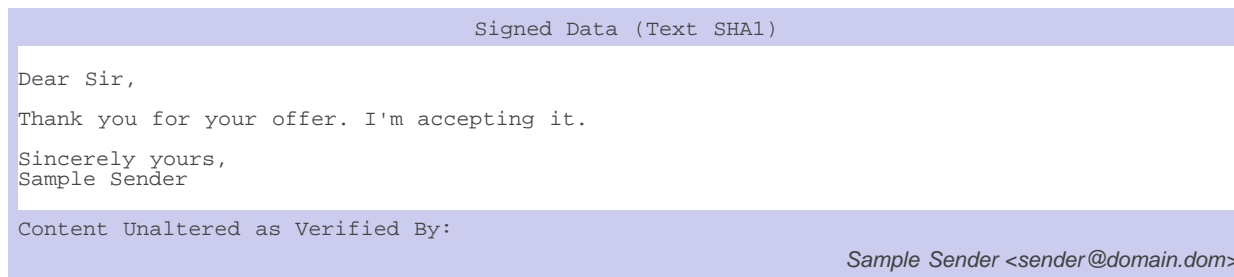
| Settings: Secure Mail | General  Password  Folders  Compose  Calendar  Contacts  Secure Mail  Public Info |
|---|---|
| **Secure Mail is Locked** | Secure Mail Password |

# Receiving Signed Messages

A message stored in your Mailbox or a message part can be digitally signed. When you open such a message, the WebUser Interface component automatically checks the integrity of the signed part. It retrieves the Signers data from the signature data and tries to verify the signature of all signers. It then shows the list of all signers whose signatures match the message content:

```
                        Signed Data (Text SHA1)

Dear Sir,

Thank you for your offer. I'm accepting it.

Sincerely yours,
Sample Sender

Content Unaltered as Verified By:
                                         Sample Sender <sender@domain.dom>
```

If the information cannot be verified with any signature, an error message is displayed.

# Recording Certificates

A Signed message contains the Certificate of the signer. The Take Certificate button that appears on the Message page when a Signed message is displayed. By clicking that button you include the E-mail address and the name of the signer (as specified in the Certificate, not in the message headers), and the signer certificate into your currently selected [Address Book](#).

When an Address Book is displayed, the `[@]` marker indicates the entries that have known (stored) certificates. You can send encrypted messages to those addresses.

# Sending Signed Messages

To Send a signed message, make sure that your Private Key is unlocked. If it is unlocked, you will see the Send Signed checkbox on your Compose page. Select this checkbox to sign your message. If you send a message with attachments, the entire content of your message, including all attachments, will be signed with your Private Key and your Certificate will added to the message signature.

Recipients of your Signed message will be able to verify that the content has not been altered, and they will be able to store your Certificate and later send you encrypted messages.

# Sending Encrypted Messages

To Send an encrypted message, make sure that your Private Key is unlocked, and that all message recipients are included into your Address Book, and their Address Book entries contain certificates.

If your Private Key is unlocked, you will see the Send Encrypted checkbox on your Compose page. Select this checkbox to encrypt your message. If you send a message with attachments, the entire content of your message, including all attachments, will be encrypted with the recipients Public Keys (taken from their Certificates), and with your own Public Key. As a result, if a copy of the encrypted message is stored in your Sent Mailbox, you will be able to read (decrypt) it.

If you select both Send Signed and Send Encrypted options, the message will be composed as a Signed message, and then the entire content (including the message headers and your signature) will be encrypted.

Use the Encryption Method WebUser setting to specify the encryption "cipher".

---

# Receiving Encrypted Messages

When you receive an encrypted message, its content is not displayed:
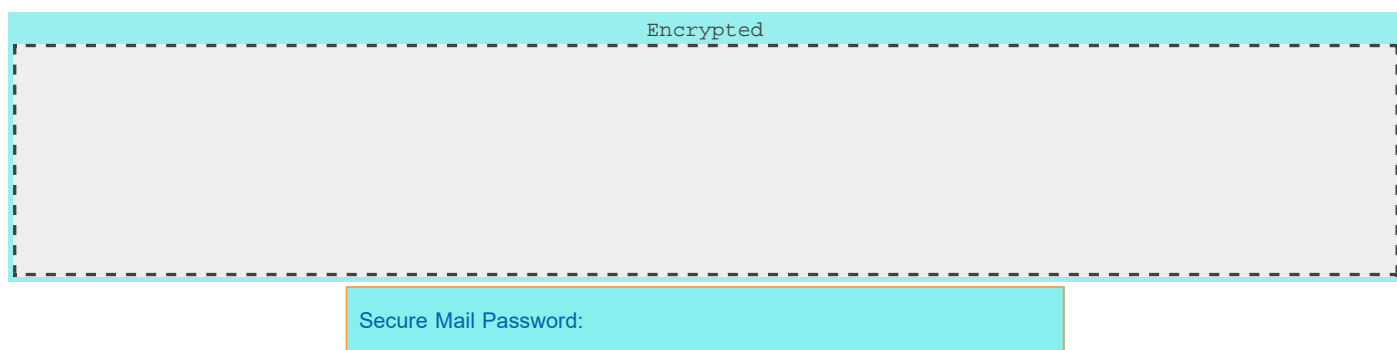


You need to activate (unlock) your Private Key first. With the Private Key unlocked, the WebUser Interface module tries to decrypt all encrypted messages with your Private Key. If it succeeds to decrypt the message, the message content is displayed:
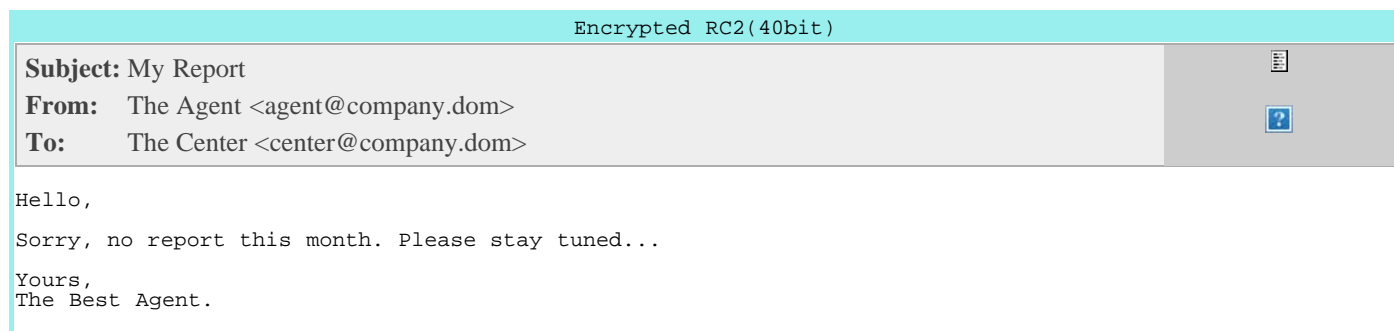


When you want to keep the message in your Mailbox, but you want to keep it in the decrypted form, click the Decrypt button. The Server will try to decrypt the encrypted message. If it succeeds, it will store the decrypted message in your Mailbox and will remove the original encrypted message.

---

# Encrypting Stored Messages

When you receive an unencrypted message, you may want to encrypt it in your Mailbox. Activate (unlock) your Private Key, and click the Encrypt button. Your Private Key is not used for encryption (the Public key from your Certificate is used), unlocking the Private Key is needed only to prove that you will be able to decrypt the message after it is encrypted.

---

# Encrypting Incoming Messages

You can automatically encrypt certain messages coming to your Account. See the [Rules](#) section, the `Store Encrypted` action.

| **Introduction** |
| **Installation** |
| **SysAdmin** |
| **Network** |
| **Objects** |
| **Storage** |
| **E-mail** |
| **Real-Time** |
| **Access** |
| **Services** |
| **Directory** |
| **Clusters** |
| **Applications** |
| **Miscellaneous** |
| **Licensing** |
| |
| **WebMail** |
| **Pronto!** |
| **PBX** |

| **Pronto!** | **Module API** | **History** |

# Pronto! XIMSS Client

- **Functionality**
- **Auto Login/Single Sign On**
- **Customization**
- **Push Notifications for Pronto! Mobile**

The Pronto! client is a client application providing access to CommuniGate Pro Accounts using the [XIMSS](#) protocol.

The Pronto! client is implemented using the Adobe® Flash technology. It is available as a zero-install "Web 2.0" client, and as a desktop "Adobe AIR" application.

If the WebCal [Service](#) is enabled for your Account, you can use the Pronto! client to create Events (Meetings and Appointments) and ToDo items (Tasks), to accept and cancel them, to view your Calendar and ToDo lists.

If the Signal [Service](#) is enabled for your Account, you can use the Pronto! client to exchange IM and Presence information with others.

If the PBX [Service](#) is enabled for your Account, you can use the Pronto! client to originate and accept phone calls and perform other Telephony operations.

# Functionality

The client provides you access to all CommuniGate Pro "services" enabled for your Account and Domain:

Mail
    If this Service is enabled, you can read and compose new E-mail messages.
WebCal
    If this Service is enabled, you can review and update your Calendar information.
Signal
    If this Service is enabled, you can send and receive Instant Messages and Presence information.
WebSite
    If this Service is enabled, you can manage your [File Storage](#).
PBX
    If this Service is enabled, you can make and receive phone calls.

# Auto Login/Single Sign On

The Pronto! client can be easily integrated into a "portal" environment, allowing a user to enter a Pronto! session

without explicitly entering the username and/or password data.

To start Pronto!, direct the browser to the following URL:

```
[http|https]://server[:port]/Pronto/[?parameters]
```

where:

`server`
　　the CommuniGate Pro server/cluster address
`parameters`
　　URL parameters. The following parameters are supported:

　　`username`
　　　　the user Account name (for Auto Login).
　　`password`
　　　　the user Account password (for Auto Login). This is not a secure Auto-Login method, and it should be
　　　　avoided.
　　`sid`
　　　　a SessionID for a [WebUser](#) or [XIMSS](#) session the specified user Account has already opened. It will be
　　　　used for Auto-Login.
　　`hideLogin`
　　　　if this parameter value is `yes`, and the `username` and `password`/`sid` parameters are specified, the Pronto!
　　　　Auto Login will take place without displaying the Pronto! Login screen.
　　`useHTTPBinding`
　　　　if this parameter value is `yes`, the Pronto! XIMSS session will be established using [HTTP binding](#) instead
　　　　of a direct TCP connection.

# Customization

## Customizing logos

The image files contained in the `Pronto4` skin can be customized to add custom logos or login scree pictures.

`loginlogoimage.png`
　　The main logo on the login page. Size: 350x117 pixels, background: transparent.
`loginimage.png`
　　Login page "screenshot image". Size: 440x340 pixels, background: opaque.
`logosmallimage.png`
　　The main screen logo in the top left corner. Size: 90x30 pixels, background: transparent.
`customlogosmall.png`
　　Rotates with the CommuniGate Pro logo in the right bottom corner of the main screen. Size: 90x30 pixels,
　　background: transparent.
`customlogomedium.png`
　　Rotates with the CommuniGate Pro logo on the login page. Size: 145x50pixels, background: transparent.
`customlogo125x37.png`
　　Rotates with the CommuniGate Pro logo on the login page. Size: 125x37pixels, background: transparent.
`loginlogo110.png`
　　The logo used in the AIR-based version of Pronto4

## Customizing preloaders

Pronto preloaders are simple `.swf` files which are executed during application initialization and can be created with
Flash Professional easily. Two templates are provided in the [archive](#). One of the sample preloaders includes logo and
displays progress as a text. The second one displays no text and no logo, so it can be used without modifications if you
need to get Pronto! installation without the Pronto logo.

The [archive](#) also includes the basic ActionScript code that make preloaders tick. Any preloader should have at least a

public property progress (of a numeric type). Optionally, a preloader can have public property `loadingText`, which is a `String`. These two properties are already used in the `Preloader.as` script in example preloaders.

To replace Pronto preloader, the custom `preloader.swf` should be installed into Pronto4 skin of the domain.

## Creating color schemes

Pronto4 provides the built-in color scheme editor. To start it right click on the Preferences button to select the "Show customization panel" command. The workflow is as follows:

1. The work in progress can be saved at any time. The design is stored in the account file storage.
2. The color scheme being designed can be downloaded using the `Download` button. The resulting file can be uploaded to the Pronto4 skin to be immediately available to the users.
3. The name of the scheme can be made more user-friendly and ready for localization process. The user-friendly names are retrieved from the `LookLabels` dictionary in `strings.data` of the Pronto4 skin. If the custom Pronto4 skin does not have yet a custom `strings.data` file, one needs to be created with just the `LookLabels` dictionary inside:

```
LookLabels = {
   special_scheme = "For color blind";
   spec2 = "High Contrast";
};
```

4. If the new scheme should be used by default, rename it to `strings.look-default.data` and upload it to the skin

## Customizing toolbar icons

The toolbar icons are stored inside the `mailloadassets.zip` archive in Pronto4 skin as plain `.png` files. To override default ones, simply download the archive from the skin, store modified icons into it and upload back to the skin.

## Adding buttons to taskbar

You can add multiple buttons to Pronto! taskbar that point to external pages, they will show up near Preferences/Logout buttons. To add them, you need to modify two dictionary parameters in `strings.data`:

`AdditionalProntoButtons`
    Specify buttons to add. The entry name is displayed on the button, the value is an URL used as a link.
`AdditionalProntoButtonNames`
    (optional) Specify localized labels for custom buttons. Entry names should correspond to entries in `AdditionalProntoButtons`, and the value is the localized button label.

URLs specified in `AdditionalProntoButtons` can contain the following templates, which are substituted by Pronto! when user clicks on the button:

`%%ID%%`
    Replaced by sessionID
`%%accountName%%`
    Replaced by accountName, no domain part
`%%domainName%%`
    Replaced by domain of the account
`%%fullAccountName%%`
    Replaced by full account name: user@domain
`%%realName%%`
    Replaced by user's real name (taken from `<session/>` XIMSS response)

## Customizing File Sharing URL Scheme

When a user shares a file in his file storage (this also happens when a user sends a file over IM), the link to the file is based on the user session properties: if the user is connected over HTTPS, `https://` link will be created, otherwise Pronto! will create `http://` link. If the user is connected over unusual port (like `8100`), it'll be also added to the link.

Sometimes it is desired to specify URL scheme and port that should be used for creating these links regardless of the user's session properties. To do so, you need to specify the following parameters (both are optional) in `strings.data`:

FileSharingURLScheme
> URL scheme to be used in shared file links. Either `http` or `https`

FileSharingURLPort
> Port number to be used in shared file links

## Customizing login page

The following parameters in `strings.login.data` skin file control the appearance of Pronto AIR login page:

ShowSecureSelector
> Set to "YES" to enable display of the "secure" checkbox.

UseSecureConnection
> Set to "YES" to force secure connection to the server when the option to select the secure connection mode is hidden.

EnableCustomSignUp
> Set to "YES" to use a custom sign-up page.

CustomSignUpURL
> Set to the string with the URL to use as the custom sign-up page.

ShowLanguageSelector
> Set to "YES" to enable language selector on the login page.

```
ShowSecureSelector = YES;
UseSecureConnection = YES;
EnableCustomSignUp = YES;
CustomSignUpURL = "https://sign-up.mydomain.com/";
ShowLanguageSelector = NO;
```

## Customizing single-window mode

The following parameter in `strings.shell.data` skin file control the appearance of main Pronto view in single-window mode:

SingleWindowModeIMPosition
> If specified and value is "left" IM panel would be placed at the left, otherwise at the right side of the view.

# Push Notifications for Pronto! Mobile

The Pronto! Mobile applications installed on iOS and Android devices support Push notifications which could be sent by the CommuniGate Pro server via APNs (Apple Push Notification Service) and FCM (Firebase Cloud Messaging) servers. When Instant Message or incoming call is directed to the user Account, the CommuniGate Pro server checks if there are any APNs or FCM tokens of the Pronto! Mobile application stored on the server for this Account and sends the Push notification requests to the APNs and FCM servers for all found tokens of the Account iOS and Android devices.

The Push notification received by the Pronto! Mobile application is displayed on the user mobile device and contains brief information about the pending call or Instant Message, such as sender Name or Address and a part of the message body. The Push notifications allow Pronto! Mobile application to wake up if it was inactive and immediately receive the incoming call or Instance Message.

To be able to send Push notification requests the APNs provider certificate and FCM server key are required. Please

contact CommuniGate Systems technical support to request these key and certificate.

Open the Preferences page of the Account Settings:

| Pronto! Preferences | | |
|---|---|---|
| Incoming Calls: default(Enabled) | Outgoing Calls via: default(Dialer) | |
| IM Audio Alert: default(Enabled) | Send Push Notifications: Enabled | |

Send Push Notifications

If this option is enabled, the server sends Push notification requests for each incoming call or Instant Message directed to this Account. Otherwise, Push notifications are not sent.

| Pronto! | **Module API** | History |

# Pronto! Module API

- **ProntoCore library**
- **ProntoCore requests and messages**
- **Providing requests to other modules**
- **Managing modules**
- **Default Pronto modules**
  - `E-Mail module`
  - `IM module`
  - `Calendar module`
  - `Telephony module`
  - `Contacts module`
  - `News module`
  - `Music module`
  - `Video module`
  - `Files module`
- **Extension points and extensions**
- **Default extension points**
  - `MainNavigator`
  - `Prefs`
  - `NewButton`
  - `FileEditors`
  - `SharingMethods`
- **Preferences**
- **Accessing XIMSS channel**
- **Strings**
- **Properties**
- **Log**
- **User idle / present**
- **ProntoCode data format**
- **Quick Start Guide**
- **Debug custom module with Flex SDK 4.6.0**
- **Creating module**
- **Debugging module: debugconfig.xml**

The Pronto! Client API is used to extend the application functionality. You can add new modules alongside the Email, Calendar, News and other already existing Pronto! modules.

See the complete Pronto! API documentation for mode details and module examples.

# ProntoCore library

ProntoCore is an ActionScript 3.0 library and provides means to exchange data with Pronto! and CGP server (using existing XIMSS channel). More information about library structure can be found in ProntoCore ActionScript Documentation (ASDoc).

# Modules lifecycle

The Main logical element of Pronto API is "Module". A Module consists of several files in server Skins:

- `mymodule.swf` - a binary Adobe Flash file
- `mymodule.def` - XML text with module metadata
- other skin files, such as images

When a user logs into Pronto!, the `.def` files in current Skin are listed and non-empty well-formed ones are treated as Modules.

All on-login (see `.def`) Modules are loaded, initialized and launched. On-demand modules are loaded during the session when the Pronto! core or some other loaded Module requests them.

When a user logs out of Pronto!, all Modules are unloaded, custom requests are unregistered and extensions/extension points are removed.

---

# ProntoCore requests and messages

This section describes principles of ProntoCore library. ProntoCore provides several means of interaction: requests and messages. Since ProntoCore is ActionScript 3.0 library all examples are written in ActionScript 3.0.

## Instant requests

Instant request is low-typed function call. Result or error (if any) is available immediately after request execution.

```
var error:Error = new Error(null);
// Retrieve information about selected module - mail, calendar or other one
var selectedExtension:Object = ProntoShell.instantPoll("getSelectedExtension",
    "MainNavigator", error);
// Every request can be executed with error
if (error.message)
    errorTextField.text = error; // show error to use
// Output name of selected section
trace("Selected tab in MainNavigator: " + selectedExtension.name);
```

Instant request can be made using `ProntoShell.instantPoll()` in case we need the request result. If we do not need the result or call do not provide any result it is more suitable to use `ProntoShell.instantSend()`:

```
var error:Error = ProntoShell.instantSend("setSelectedExtensionIndex",
    { extensionPointName: "MainNavigator", index: 0 });
// Every request can be executed with error
if (error.message)
    errorTextField.text = error; // show error to user
```

## Synchronous requests

Synchronous request is low-typed delayed function call. When synchronous request is made, caller may specify two callback functions, that will be called when data and/or response arrives.

Synchronous Pronto API requests work the same way XIMSS synchronous requests do.

```
ProntoShell.syncPoll("ximss", <fileDirInfo/>, dataCallBack,
    responseCallBack);
```

```
function dataCallBack(xml:XML)
{
    trace("fileDirInfo data response: " + xml);
}

function responseCallBack(object:Object)
{
    var text:String;
    if (object is Error)
        text = Error(object).message;
    else if (object is XML && XML(object).hasOwnProperty("@errorText"))
        text = XML(object).@errorText;
    if (text)
        errorTextField.text = "fileDirInfo has failed: " + text;
}
```

If synchronous request do not provide any data messages it is more suitable to use `ProntoShell.syncSend()`:

```
ProntoShell.syncSend("ximss", <fileRename fileName="tasks.txt" newName="oldTasks.txt"/>, responseCallBack);

function responseCallBack(object:Object)
{
    var text:String;
    if (object is Error)
        text = Error(object).message;
    else if (object is XML && XML(object).hasOwnProperty("@errorText"))
        text = XML(object).@errorText;
    if (text)
        errorTextField.text = "fileRename has failed: " + text;
}
```

## Asynchronous messages

Asynchronous message is a message, that can arrive at any time.

```
ProntoShell.dispatcher.addEventListener("ximss-readIM", readIMHandler);

function readIMHandler(event:Object):void
{
    var xml:XML = event.xml;
    trace(xml.@peer + " said: " + xml.toString());
}
```

```
ProntoShell.dispatcher.addEventListener("mainNavigatorSelectionChange",
    selectionChangeHandler);

function selectionChangeHandler(event:Object):void
{
    // event object has properties oldIndex, newIndex, oldExtension, newExtension
    var extension:Object = event.newExtension;
    trace("Selected extension name: " + extension.name);
}
```

*Note: unsibscribe from asynchronous events using* `removeEventListener` *on "moduleUnloading" event with* `event.moduleId == "MyModule"`. ***Otherwise your module would not be collected by Garbage Collector and objects will recieve events after relogin.*** *Using weak reference does not always help.*

For more information on ProntoCore library see it's ASDoc.

---

# Providing requests to other modules

Pronto modules can create their own instant, synchronous requests and asynchronous messages. It can be done via static methods `ProntoShell.instantRegister`, `ProntoShell.syncRegister` and via dispatching events through static property `ProntoShell.dispatcher` (`IEventDispatcher`).

**Note:** All requests, registered by modules, are unregistered on logout.

**Note:** If you publish module that provides public request, make sure to maintain backward-compatibility for that requests.

## Custom instant request

**Example:** Register `getLatestNews` instant request, that provides access to some internal module data

```
ProntoShell.instantRegister("getLatestNews", getLatestNewsHandler, "News");

function getLatestNewsHandler(request:String, param:Object, error:Error):String
{
    var timePeriod:int = param.hasOwnProperty("timePeriod") ? param.timePeriod : 0;
    if (timePeriod > 0)
    {
        var result:String;
        // custom logic, setting result to some value
        return result;
    }
    else
    {
        // provide error message
        error.message = "timePeriod property is not specified or is incorrect";
        return null;
    }
}
```

## Custom synchronous request

**Example:** register `myService` synchronous request

```
var error:Error = ProntoShell.syncRegister("myService", myServiceHandler, "MyModule");
if (error)
    throw error; // better handle it somehow

function myServiceHandler(request:String, param:Object, dataHandler:Function,
    responseHandler:Function):void
{
    var xml:XML = param as XML;
    if (!xml)
    {
        responseHandler(new Error("myService parameter should be XML"));
        return;
    }

    // dataHandler should be called before responseHandler and can be called
    // more than once. responseHandler should be called in any case. All this
    // can be done any time - right now or a minute later.
    dataHandler({ label: "Label 1" });
    dataHandler(new Date());
    responseHandler("All done!");
}
```

## Custom asynchronous message

```
// code in News module. Notify everybody, that news feeds are loaded
ProntoShell.dispatcher.dispatchEvent(new Event("newsLoaded"));
```

# Managing modules

Pronto! provides the following requests and messages that allow to retrieve information about modules, load and unload modules, etc.

## Instant requests

`moduleData`

Retrieves information about module. `Object` is returned (see [ProntoCore data](#)). Module id should be passed as parameter.

`moduleDataList`
>   Retrieves information about all modules. `Array` is returned.

`moduleHas`
>   Checks module existence. `String` module id should be passed as parameter. `Boolean` is returned.

`moduleLoad`
>   Loads on-demand loading module. `String` module id should be passed as parameter.

`moduleUnload`
>   Unloads already loaded and initialized (or failed to load or initialize) module. `String` module id should be passed as parameter.
>
>   *Note: some modules (including all Pronto default modules) do not support second load during use session. This happens because module tries to find extensions, that should be added from `.def` file, while they were removed during unload.*

`getInitialized`
>   Returns `Boolean` value describing if the whole initialization process is complete: all on-login modules are loaded (or failed to load) and initialized (or failed to initialize).

`moduleFileURL`
>   Retrieves URL for the given file.
>
>   `moduleId:String`
>   >   Id of the module.
>
>   `fileName:String`
>   >   Name of the module's file (icon or other asset).

## Synchronous

`initialized`
>   Each module should call this function after it has finished adding extension points, registering requests and other stuff, that can be used by other modules. Simple module that just adds one tab to `MainNavigator` usually makes this call first a does not even specify `responseHandler`.
>
>   Response to this request comes simultaneously to all modules when all on-login modules are initialized (or have failed to load or initialize). For on-demand loading modules request comes immediately (if all on-login loading modules are already initialized).
>
>   `moduleId:String`
>   >   Module id.
>
>   `errorText:String`
>   >   Optional fatal error description. If specified, error is logged in module errors and `status` property becomes `error` (see [module object format](#))
>
>   If `String` is passed instead of `Object`, it is treated as module id.

```
ProntoShell.syncSend("initialized", "com.myCompany.myModule.MyModule", responseCallBack);
```

## Asynchronous

`moduleInitialized`
>   This event is dispatched after Pronto! has called module `moduleId` response handler for his `initialized` request.
>
>   `moduleId:String`
>   >   Module id.

`moduleError`

This event is dispatched when module fails to load or if it reports error in `initialized` request.

`moduleId:String`
> Module id.

`errorText:String`
> Optional. Is specified if module has specified error description in `initialized` request.

moduleUnloading
> This event is dispatched before unloading module `moduleId`. At that moment no operations (like removing module extensions) are executed yet.
>
> This event can be used to desctruct module: remove extensions and extension points, unsibscribe from `ProntoShell.dispatcher`, unload data, release some data channels and etc.
>
> `moduleId:String`
> > Module id.

moduleUnload
> This event is dispatched after module `moduleId` has been unloaded, all his extension points and extension removed and other cleanup completion.
>
> `moduleId:String`
> > Module id.

---

# Default Pronto modules

## E-Mail module

E-Mail is not yet a true module, now it is an internal part of Pronto.

Adds `Mail` extension to `MainNavigator` with `preferredIndex 10`.

Adds `Folders` with `preferredIndex 20`, `Compose` with `preferredIndex 30`, `Delete` with `preferredIndex 40`, `MailControl` with `preferredIndex 90` and `RPOP` with `preferredIndex 100` to `Prefs`.

Adds `Mail` extension with `preferredIndex 10` and `MailNewFolder` with `preferredIndex 20` to `NewButton`.

**Synchronous requests**

mailCompose
> Opens mail composing popup.
>
> `toValue:String`
> > Content for the "To:" field.
> `subject:String`
> > Content for the "Subject:" field.
> `body:String`
> > Mail body.

mailShowNewFolderDialog
> Opens new mail folder popup.

addressBookSelect
> Opens Address Book and lets user select items from contacts folders.
>
> `showPhones:Boolean`
> > Defines whether phone numbers will be shown in the contacts grid or not.
> `allowMultipleSelection:Boolean`

Optional. If set to `true`, multiple selection is allowed.

Data handler receives object with `items:Array` property, that contains objects with the following properties:

`subject:String`
      Textual description of contact or contact group.
`toValues:Array`
      Optional. Not empty array of `String` user E-Mail addresses and phones (if `showPhones` was set to `true`).
`selectedItem:String`
      Optional. If user has picked one item from the list of `toValues`, it is specified here.

*Note: Pronto allows only one instance on the Address Book. If it is called (ex. from Dialer) during* `addressBookSelect` *execution, request finishes execution without data just like when user closes Address Book popup.*

# IM module

IM is not yet a true module, now it is an internal part of Pronto.

Adds `IM` extension with `preferredIndex 50` to `Prefs`.

**Synchronous requests**

`imAdd`
      Shows adding IM contact dialog with pre-filled fields. If given `peer` already exists all other properties are ignored and popup opens in edit mode.

`peer:String`
      Optional. Peer, usually an E-Mail address.
`realName:String`
      Optional. Real name.
`groups:Array`
      Optional. User groups. Pronto yet supports only 1 group for user, so the first array item is used.

# Calendar module

Calendar is not yet a true module, now it is an internal part of Pronto.

Adds `Calendar` extension to `MainNavigator` with `preferredIndex 20`, to `Prefs` with `preferredIndex 70` and to `NewButton` with `preferredIndex 20`.

**Synchronous requests**

`calendarShowNewEventDialog`
      Shows "Event" dialog to create new event

# Telephony module

Telephony is not yet a true module, now it is an internal part of Pronto.

Adds `Telephony` extension to `MainNavigator` with `preferredIndex 30`, `CallControl` extension with `preferredIndex 110` and `Dialer` extension with `preferredIndex 120` to `Prefs`.

**Instant requests**

getPhoneStatus
     Returns `Object` with properties:

     `active:Boolean`
          Is `true` if phone is enabled and has at least one active call.
     `enabled:Boolean`
          Is `true` if phone was initialized and is functioning.
     `calls:Array`
          Array of objects with properties `peer:String` with peer address and `status:String` with description of
          call status. Status should not be interpreted.

**Asynchronous messages**

phoneStatus
     Is dispatched when phone status changes. This includes enabling / disabling, calls and their statuses changes.
     Event object has the same properties as `getPhoneStatus` result.

# Contacts module

Contacts are not yet a true module, now it is an internal part of Pronto.

Adds `Contacts` extension to `MainNavigator` with `preferredIndex 40` and to `Prefs` with `preferredIndex 80`. Adds
`ContactsNewContact` extension with `preferredIndex 30` and `ContactsNewGroup` extension with `preferredIndex 40`
to `NewButton`.

**Synchronous requests**

contactsShowNewContactDialog
     Shows "Contact" dialog to create new contact.

contactsShowNewGroupDialog
     Shows "Contact Group" dialog to create new contact group.

# News module

Module id is `News`.

Adds extension `News` to `MainNavigator` and `Prefs`.

# Music module

Module id is `Music`.

Adds extension `Music` to `MainNavigator` and `Prefs`.

Adds `MusicPlayerEmbed` and `MusicPlayerLink` extension to `SharingMethods`.

# Photos module

Module id is `Photos`.

Adds extension `Photos` to `MainNavigator`.

# Video module

Module id is `Video`.

Adds extension `Video` to `MainNavigator`.

## Files module

Module id is `Files`.

Adds extension `Files` to `MainNavigator` and `FilesNewDirectory` to `NewButton`.

Defines [FileEditors](#) extension point.

Adds extensions to `FileEditors` extension point: `TextEditor` with `preferredIndex 10`, `ImageViewer` with `preferredIndex 20`, `MP3Player` with `preferredIndex 30`, `HTMLViewer` with `preferredIndex 40` and `VideoPlayer` with `preferredIndex 50`.

### Synchronous requests

`filesShowNewDirectoryDialog`
> Shows "Create Directory" dialog.

---

# Extension points and extensions

Pronto API utilizes well-known system of *extension points*, that are some logical or visual containers and *extensions*, that are items in that containers.

Modules can create new Extension Points, add, list and remove extensions from existing Extension Points and more.

Typically module define one extension for `MainNavigator` extension point in `def` file. This lets Pronto show module in `MainNavigator` without loading it. When module gets loaded, it should replace `def` file extensions to add actual content to them because for example `MainNavigator` specifies progress bar for `content` property (see examples).

## Instant requests

`extensionsList`
> Lists current extensions starting from extension with index 0. `Array` or `Object` is returned.
>
> > `extensionPointName:String`
> > > Target extension point.
> > `extensionName:String`
> > > Optional. If specified, only that extension is returned.

`extensionAdd`
> Adds new extension to existing extension point.
>
> > `extensionPointName:String`
> > > Name of extension point.
> > `extension:Object`
> > > Extension.
> > `buffer:Boolean`
> > > If `true` and extension point does not exist, extension will be cached and added as soon, as extension point will be created.
> > `replace:Boolean`
> > > It `true` and extension with specified name already exist in extension point it will be seamlessly replaced with new one.

This option is useful if you replace extension that was added using `.def` file.

Example:

```
var error:Error = new Error();
var extension:Object = ProntoShell.instantPoll("extensionsList",
    { extensionPointName: "MainNavigator", extensionName: "Files" }, error);
if (error.message)
    throw error;

var extensionObject:Object = {};
extensionObject.name = "Files";
extensionObject.moduleId = "Files";
extensionObject.preferredIndex = extension.preferredIndex;
extensionObject.localizedLabelId = extension.localizedLabelId;
extensionObject.iconFactory = extension.iconFactory;
extensionObject.contentFactory = new ClassFactory(FilesContent);
extensionObject.navigationFactory = new ClassFactory(FilesNavigation);
extensionObject.toolBarFactory = new ClassFactory(FilesToolBar);
var object:Object = {};
object.extension = extensionObject;
object.extensionPointName = "MainNavigator";
object.replace = true;
error = null;
error = ProntoShell.instantSend("extensionAdd", object);
if (error)
    throw error;
```

extensionRemove

>    Removes existing extension from extension point.

>    extensionPointName:String
>>        Name of the extension point.
>    extensionName:String
>>        Name of the extension.

extensionHas

>    Checks extension existence. `Boolean` is returned.

>    extensionPointName:String
>>        Name of the extension point.
>    extensionName:String
>>        Name of the extension.

extensionPointAdd

>    Adds new extension point. `Object` extension point should be passed as parameter (see [ProntoCore data](#)).

extensionPointRemove

>    Removes existing extension point. `Object` extension point object should be passed as parameter.

extensionPointHas

>    Checks extension point existence. `String` extension point name should be passed as parameter. `Boolean` is
>    returned.

getSelectedExtension

>    *This functionality is optional, extension point may not support selecting.* Retrieves selected extension. `String`
>    extension point name should be passed as parameter. `Object` is returned.

setSelectedExtension

>    *This functionality is optional, extension point may not support selecting.* Sets selected extension.

>    extensionPointName:String
>>        Target extension point.
>    extensionName:String
>>        Name of the extension to select.

getSelectedExtensionIndex

>    *This functionality is optional, extension point may not support selecting.* Retrieves selected extension index.
>    `String` extension point name should be passed as parameter. `int` is returned.

setSelectedExtensionIndex

>    *This functionality is optional, extension point may not support selecting.* Sets selected extension index.

```
extensionPointName:String
```
  Target extension point.
```
index:int
```
  Extension index. `-1` removes selection *(if supported by the extension point)*.

---

# Default extension points

## MainNavigator

Forms the whole Pronto! visual structure. Mail, Calendar and other sections are extensions in `MainNavigator`.

Supported extensions properties are [standard](#) plus the following.

Extension object properties

```
iconFileName:String
```
  In additional to standard functionality, more that one file can be specified, for example `"icon16.png`
  `icon32.png icon64.png"`. MainNavigator uses most suitable one at the moment.
```
navigation:IUIComponent and DisplayObject
```
  Similar to `content`, used as navigation visual object.
```
navigationFactory:IFactory
```
  Similar to `contentFactory` for navigation.
```
toolBar:IUIComponent and DisplayObject
```
  Similar to `content`, used as toolbar.
```
toolBarFactory:IFactory
```
  Similar to `contentFactory` for toolbar.

```
mainNavigatorSelectionChange
```
  Event, dispatched when selection changes. See [selection change event](#).

## Prefs

Extensions from this Extension Point form list of available panels in "Preferences" popup ("Settings" in previous versions).

Supported extensions properties are [standard](#).

`Prefs` has one difference in `content` property. Each time the Preferences popup is opened, `content` is recreated using `contentFactory`. When popup is closed, `content` properties of all extension are set back to `null`, that means you can use only contentFactory property safely.

Contains `General` extension with `preferredIndex 10` and `Password` extension with `preferredIndex 60`.

For more information about `Prefs` extension logic see [Preferences](#).

## NewButton

This Extension Point corresponds to "New..." button at the top-left of the screen. It allows to compose letter, create contact and so on. Modules can add custom buttons to dropdown menu and can bind them with `MainNavigator` extension. Ex. when `Files` extension is selected in `MainNavigator`, "New..." button will open file directory creation dialog.

Supported extensions properties are [standard](#) excluding `content` and `contentFactory` properties.

Extension object properties

```
    mainNavigatorExtensionName:String
          Name of corresponding extension in MainNavigator extension point.
    callBack:String
          Name of synchronous request that should be called when button is pressed.
```

Example: adding `FilesNewFolder` extension to `NewButton` in `files.def`

```
<extension extensionName="FilesNewFolder" extensionPointName="NewButton" mainNavigatorExtensionName="Files"
    localizedLabelId="DirectoryTag" preferredIndex="5" iconFileName="filesicon16.png"
callBack="filesNewFolderDialog"/>
```

## FileEditors

This Extension Point holds editors and viewers that are used in `Files` module.

Supported extensions properties are [standard](#) excluding `contentFactory`. Apart from them the following properties are valid:

Extension object properties

```
    fileTypes:String
          Space-separated extensions, that this viewer / editor can handle. Ex. jpg jpeg gif png for image viewer.
          Case-insensitive.
```

Moreover, `content` property should provide the following properties:

```
    edit:Function - function(object:Object):void
          This function is called when editor is added to stage and should show and/or edit given object that is File
          Storage file object.
    get changed():Boolean
          Getter method, should return Boolean indicating unsaved changes in current file. If extension is viewer
          and do not change file itself, this function should always return false.
    save:Function - function():void
          Calling this function should start file saving.
    clean:Function - function():void
          Calling this function should cause total editor cleanup (even if there are unsaved changes).
```

After editor has changed the file on the server it should dispatch `fileChanged` event so that Pronto! change internal file URLs to present the latest file version to the user.

*Note: at this moment it is not allowed to add editors through `.def` file. `content` property should be populated.*

## SharingMethods

This Extension Point contains methods for File Storage objects sharing.

Contains `URLSharingMethod` extension with `preferredIndex 10` and `FileBrowserSharingMethod` extension with `preferredIndex 20`.

Supported extensions properties are [standard](#) except `content` and `contentFactory` properties. Apart from them the following properties are valid:

Extension object properties

```
    textFunction:Function - function(object:Object):String
          Function should recieve File Strorage file or directory and return string, that will be shown to user. It can
```

be an url or an embed-code.

If File Storage object can not be shared using this method, `null` should be returned.

---

# Preferences

Pronto API allow modules to create, read and remove their own custom preferences and add panels to "Preferences" popup using requests from this section.

## Instant handlers

`getPref`
>Returns XML node with given name (if found) from settings of given type.

>>`type:String`
>>>Optional. Possible values are `custom`, `default` or `effective`. If not specified, effective settings are meant.
>>`prefName:String`
>>>Name of the XML node.

```
var error:Error = new Error(null);

var pref:XML = ProntoShell.instantPoll("getPref", { prefName: "CalendarBox", type: "effective" }, error);
if (error.message)
{
    errorTextField.text = error.message;
    return;
}
var calendarBox:String = pref ? pref.toString() : "Calendar";
```

`prefName` value can be passed as a parameter to simply get effective preference value:

```
var error:Error = new Error(null);

var pref:XML = ProntoShell.instantPoll("getPref", "CalendarBox", error);
if (error.message)
{
    errorTextField.text = error.message;
    return;
}
var calendarBox:String = pref ? pref.toString() : "Calendar";
```

`setPref`
>Adds given `prefXML` to custom preferences XML. To remove custom preference, add node with `Default` text inside. This will cause preference to remove on next update.

>New preferences are not available through `getPref` immediately. They apply only after next `prefsUpdated` event.

>Does not cause custom preferences to be sent to the server immediately.

>>`prefXML:XML`
>>>Preference itself.
>>`relogin:Boolean`
>>>Defines, whether this preferences change will be applied immediately or only on next login. Ex. skin can not be changed during the session.

```
var error:Error = ProntoShell.instantSend("setPref", { prefXML: <myOwnPref><myValue myAtt="1"/></myOwnPref>
});
if (error)
    errorTextField.text = error.message;
```

```
// removing myOwnPref
var error:Error = ProntoShell.instantSend("setPref", { prefXML: <myOwnPref>Default</myOwnPref> });
if (error)
    errorTextField.text = error.message;
```

prefsUpdate
>    Sends custom preferences to the server. `prefsUpdating` event is dispatched immediately and `prefsUpdated` is
>    dispatched on success or fail. Note that `prefsUpdated` is dispatched 2 times, for new effective and custom
>    preferences (in this order).

prefRelogin
>    Returns `Boolean` indicating whether there are changes in custom preferences that can be applied only on the next
>    login.

prefsOpen
>    Opens "Preferences" popup.
>
>    selectedExtensionName:String
>>        Optional. Name of extension to be selected.

prefsLoaded
>    `String` type can be specified as parameter with possible values `default`, `custom` and `effective` (used by
>    default).
>
>    Returns `Boolean` indicating whether Pronto! has already loaded preferences with a given type.

## Asynchronous messages

prefsUpdating
>    Is dispatched when custom preferences are being sent to the server. Is typically used to disable preferences-
>    controlling inline (not in "Preferences" popup) controls until `prefsUpdated` event arrive.

prefsUpdated
>    Is dispatched when any preferences XML (custom, default or effective) arrive from the server.
>
>    xml:XML
>>        Optional. Recieved XML itself. Is defined if `errorText` is not defined.
>    errorText:String
>>        Optional. Error during saving custom preferences to the server. Is defined if `xml` is not defined.

prefsSave
>    Is dispatched when user presses "Modify" button in "Preferences" popup. When custom preferences tab receives
>    this event it should commit all changes that user have made to Pronto! using `setPref` request.

prefsClose
>    Dispatched when "Preferences" popup is closed by user or on logout. **Custom preferences tabs should listen
>    this event to destruct itself to avoid problems on next Preferences popup opening and saving.**

See also: MyModule example.

# Accessing XIMSS channel

ximss
>    Synchronous ProntoCore request. Sends synchronous XIMSS request. `XML` should be passed as parameter.
>
>    *Note: you shouldn't set `id` attribute, it is set automatically.*

```
ProntoShell.syncPoll("ximss", <fileDirInfo/>, dataCallBack,
    responseCallBack);

function dataCallBack(xml:XML)
{
```

```
      trace("fileDirInfo data response: " + xml);
}

function responseCallBack(object:Object)
{
    var text:String;
    if (object is Error)
        text = Error(object).message;
    else if (object is XML && XML(object).hasOwnProperty("@errorText"))
        text = XML(object).@errorText;
    if (text)
        errorTextField.text = "fileDirInfo has failed: " + text;
}
```

ximss-...

>   Asynchronous ProntoCore message. Is dispatched each time any asynchronous XIMSS message arrive.

>   xml:XML
>>      XIMSS XML data.

>   Example:

```
ProntoShell.dispatcher.addEventListener("ximss-readIM", readIMHandler);

function readIMHandler(event:Object):void
{
    var xml:XML = event.xml;
    trace(xml.@peer + " said: " + xml.toString());
}
```

# Strings

## Instant requests

getStringsXML
>   Returns current strings `XML`.

getString
>   `String` identifier should be passed as parameter. It can be dot-separated, ex. `HeaderNames.EMAIL` corresponds `E-mail` string. If value is not found, identifier itself is returned.

## Asynchronous messages

stringsChange
>   Is dispatched each time `getStringsXML` call result changes.

# User idle / present

## Instant requests

getUserPresent
>   Returns `Boolean` specifying if user is idle or present based on AIR `NativeApplication.timeSinceLastUserInput` property and `AwayTimer` preference. If Pronto! runs not as AIR Application, always returns `true`.

## Asynchronous messages

userIdle
>   Is dispatched when user is treated as idle. Available only in AIR version.

userPresent
>   Is dispatched when user is treated as present. Available only in AIR version.

# Properties

The properties mechanism provides access to common data like session ID, user name and many more. Custom modules can provide their own properties.

## Instant requests

getProp

> `String` property name should be passed as parameter. Returns `Object` property value or `undefined` if it does not exist.

```
var error:Error = new Error();
var sessionId:String = ProntoShell.instantPoll("getProp", "sessionID", error) as String;
```

setProp

> Sets property to a specified value. Parameter should have the following properties:

> name:String
>> Name of the property. It is recommended that custom modules use package-like names for their properties, ex. `com.myCompany.myModule.myProperty1`

> value:Object
>> Value of the property.

```
ProntoShell.instantSend("setProp", { name: "com.myCompany.myModule.myProperty1", value: 41 });
```

getProps

> Returns `Object` with all properties.

## List of properties

sessonID:String
> Session id.

userName:String
> User name from `<session/>` server response.

realName:String
> User name from `<session/>` server response.

secure:Boolean
> Is connection secure or not.

host:String
> Current host.

knownValues:XML
> `<knownValues/>` node. Example:

```
<knownValues id="2">

  <tzid name="HostOS"/>
  <tzid name="(+0100) Algeria/Congo"/>
  <tzid name="(+0200) Egypt/South Africa"/>
  ...

  <charset name="iso-8859-1"/>
  <charset name="windows-1252"/>
  ...

  <mailRuleAllowed name="All But Exec"/>

  <mailRuleAllowed name="Any"/>
  ...

  <mailRuleCondition name="From"/>
  <mailRuleCondition name="Sender"/>
  ...

  <mailRuleAction name="Store in" allowedSet="Filter Only"/>
  <mailRuleAction name="Mark" allowedSet="Filter Only"/>

  ...
```

```
    <signalRuleAllowed name="No"/>
    <signalRuleAllowed name="Any"/>
    ...

    <signalRuleCondition name="Method"/>
    <signalRuleCondition name="RequestURI"/>
    ...
</knownValues>
```

# Log

The following functionality allows modules to use the Pronto! Log.

## Instant requests

`log`
> Adds message to log.
>
> `message:String`
>> Log message.
>
> `component:String`
>> Component identifier. Usually it is module id.
>
> `level:String`
>> Level of log message. Possible values are `debug`, `info`, `error` and `fatal`.

```
ProntoShell.instantSend("log", { message: "Starting initialization",
    component: Model.MODULE_ID, level: "info" });
```

# ProntoCore data format

.def file format
> This file is stored in server Skins and holds module metadata. Root tag is `module`.
>
> Attributes:
>
> `moduleId`
>> Unique id of the module
>
> `fileName`
>> Name of corresponding binary .swf
>
> `version`
>> Not interpreted, is used during composing URLs to module files to avoid caching
>
> `loadType`
>> `onLogin` (loaded immediately after login) or `onDemand` (loaded on explicit call - when module is selected in some Extension Point or "moduleLoad" request made)
>
> Body:
>
> `requirements`
>> Optional. List of options, that should be enabled for account (see `currentStatus` XIMSS response). Contains `requirement` nodes with `option` attribute.
>
> `syncHandlers`
>> Optional. Contains child nodes `syncHandler` with attribute `name`. List of synchronous handlers that module will register. Meaningful when module `loadType` is `onDemand`. List is used to allow clients make synchronous requests without checking if this module is loaded or not. If Pronto API receives synchronous request from this list and this module is not loaded, the module loading starts and request is executed when the module is loaded. If module loading fails, request is finished with the corresponding error code.
>
> `extensions`
>> Optional. Contains child nodes `extension`. Meaningful when module `loadType` is `onDemand`. List of

extensions, that should be added to extension points on login. Each `extension` node is transformed to `extensionAdd` request using the following rules:

- `extensionPointName` attribute - name of target extension point
- `extensionName` attribute - name of this extension
- All other `extension` node attributes are moved to request extension object unchanged and without interpreting
- **Note:** do not specify `moduleId`, it is set from root XML tag

Example:

```
<module moduleId="com.pronto.modules.video" fileName="video.swf" version="1" loadType="onDemand">

        <requirements>
                <requirement option="HTTP"/>
        </requirements>

        <syncHandlers>
                <syncHandler name="loadVideos"/>
        </syncHandlers>

        <extensions>
                <extension extensionName="Video" extensionPointName="MainNavigator" preferredIndex="11"
                        localizedLabelId="MyVideos" iconFileName="videoicon16.png videoicon24.png
videoicon32.png videoicon48.png"/>
        </extensions>

</module>
```

Module object.

This object can be retrieved using `moduleData` and `moduleDataList` requests.

`moduleId:String`
    Unique module id.
`version:String`
    Module version.
`status:String`
    Possible values:

- `notLoaded`
- `loading`
- `error`
- `initializing`
- `initialized`

`fileName:String`
    Module .swf file name.
`requirements:Array of String`
    Array of module requirements.
`syncHandlers:Array of String`
    Array of module synchronous handlers
`extensionAddRequests:Array of Object`
    Array of extension add requests
`errors:Array of Error`
    Errors reported by module
`bytesLoaded:int`
    Loaded bytes of the `.swf`.
`bytesTotal:int`
    Size of module `.swf`.

Extension object

Pronto API allow custom extension points, so it does not put over any restrictions on extension object properties except `name` and `moduleId`. They are **the only obligatory properties of extension object.** That properties and other commonly used in standard extension points extension properties are listed below.

`name:String`
Name of the extension, unique in extension point.
`moduleId:String`
Id of module that has added this extension. When module is unloaded, all his extensions are removed.
`content:IUIComponent and DisplayObject`
Content that should be added, for example `mx.containers.VBox` instance.
`contentFactory:IFactory`
Factory to create `content` if it is absent.

*Note: content will be created only when it is actually needed. Using factories increases performance.*

`preferredIndex:int`
All extensions in point are sorted by this index.

*Note: `getSelectedExtensionIndex` returns real index, not this one*

`label:String`
Extension label. Visual representation depends on extension point. Has lower priority than `localizedLabelId`
`localizedLabelId:String`
Id of localized string that should be used as a label. Use "." to separate inherited elements.

Example: the following part in strings XML

```
<strings skinName="Pronto-">
        ...
        <strings name="ACLRights">
                <string name="a">Admin</string>
                ...
        </strings>
</strings>
```

can be addressed as `ACLRights.a`

*Currently available localized labels and their ids can be viewed on your CGP server, for example at /SkinFiles/mydomain.com/\*/strings.xdata*

`iconFactory:IFactory`
Factory should produce `DisplayObject` instances that will be treated as icons.

**Note:** Pronto loads icons for on-demand modules, so when actual module code is loaded it can reuse the existing `iconFactory` value

`iconFileName:String`
Name of image in server Skins. Checked if `iconFactory` is not defined.

Extension point object.
Each function is called during executing corresponding instant request. `param` is passed untouched.

`name:String`
Unique name of the extension point.
`extensionsList:Function functionName(param:Object, error:Error):Array`
Lists current extensions.
`extensionAdd:Function functionName(param:Object, error:Error):void`
Adds new extension to extension point. `param` is passed untouched.
`extensionRemove:Function functionName(param:Object, error:Error):void`
Remove extension. `param` is passed untouched.
`extensionHas:Function functionName(param:Object, error:Error):Boolean`
Check extension existence. `param` is passed untouched.
`remove:Function functionName(param:Object, error:Error):void`
Remove extension point. Is called when extension point is removed (for example, on logout). Should not remove extensions and dispatch events about their removal. Instead of it whole extension point must be

removed.

`getSelectedExtension:Function functionName(param:Object, error:Error):Object`
> *Optional. Some extension point do not support selection due their nature.* Retrieves currently selected extension, `null` is nothing is selected.

`setSelectedExtension:Function functionName(param:Object, error:Error):void`
> *Optional. Some extension point do not support selection due their nature.* Sets selection to a given extension, `null` to deselect.

`getSelectedExtensionIndex:Function functionName(param:Object, error:Error):int`
> *Optional. Some extension point do not support selection due their nature.* Retrieves currently selected index, `-1` if nothing is selected.

`setSelectedExtensionIndex:Function functionName(param:Object, error:Error):void`
> *Optional. Some extension point do not support selection due their nature.* Sets currently selected index, `-1` to deselect.

## Extension Point selection change event properties

This asynchronous event is dispatched when some Extension Point selection changes. This can happen when user, module or Pronto change selected extension, remove or add new extension.

`oldExtension:Object`
> Previously selected extension. `null` if nothing was selected.

`newExtension:Object`
> Currently selected extension.

`oldIndex:int`
> Previously selected index.

`newIndex:int`
> Currently selected index.

## File Storage object

This is description of abstract File Storage object. This object is used to describe common properties of file and directory in File Storage. Each File Storage file object and File Storage directory object has these properties:

`type:String`
> `directory` or `file`

`name:String`
> Name of the object, ex. `myphotos` or `john.jpg`

`directory:String`
> Parent directory, ex. `private/myphotos`

`path:String`
> Full object path, ex. `private/myphotos/john.jpg`

`size:int`
> Size of the object

`publicURL:String`
> Optional. Is available if file or directory is not inside `private` directory or if it has special `.meta` file with key.

`sessionURL:String`
> URL to the object that is valid during current XIMSS session.

## File Storage file

File Storage file has the following additional properties:

`extension:String`
> Extension of this file, ex. `mp3`. If file does not have extension - `null`.

## File Storage directory

File Storage directory has the following additional properties:

`publicFileBrowserURL:String`
> Optional. Is available if file or directory is not inside `private` directory or if it has special `.meta` file with key. URL to the page, that shows directory content and allow to navigate inside it. URL targets `filebrowser.wcgp`.

# Creating Module

For a complete step-by-step module creating instructions see [Quick Start Guide](). This section will provide only module requirements:

1. Module should be created using **Flex SDK 3.2.0**.

   *Note: Flex Builder 3 use older version of SDK by default, you have to make it use Flex SDK 3.2.0.*

2. Module root tag should be `<mx:Module>` since it is loaded as Flex Framework module.
3. When module .swf is loaded into Pronto, all module classes with the names (`com.package.ClassName`) that already exist in Pronto will be discarded and Pronto! classes will be used instead. This causes the following issues:
   - Module should use the same version of SDK as Pronto uses.
   - Module can exclude ProntoCore and Flex SDK from `.swf` to reduce file size
   - Module can not make changes to classes that exist in Pronto or Flex SDK (ex. change `mx.controls.Button`) because those changes will be discarded.
   - Module should not include custom classes inside the following packages, that are used by Pronto:
     - com.pronto.*
     - com.stalker.*
     - com.adobe.*
     - com.flextoolbox.*
     - com.qs.*
     - com.sho.*
     - org.gif.*
     - org.xif.*
     - Text Layout Framework packages

# Debugging module: debugconfig.xml

Pronto! loads its modules from server skins, which is far from convenient if you're developing a module. In that case you have to upload each and every version you want to try inside server skin.

Test versions of Pronto! can be executed locally with the help of `debugconfig.xml` file, which should be placed near `pronto.swf` on your hard drive.

`debugconfig.xml` is intended to simplify the process of module development and makes it possible to run web version of Pronto! on developer's computer. It allows setting the server Pronto! should use, adding local modules and configuring remote modules to be loaded from local folders.

A typical `debugconfig.xml` file:

```
<debug>
    <!-- defaults: port="8100" secure="no" -->
    <server host="localhost" port="8100" secure="no" />
    <modules>
        <!-- Load .swf, and icons for module, already existing on server, from local path -->
        <module id="com.myCompany.releasedModule" basePath="com/myCompany/releasedModule" />

        <!-- Load .def, swf and icons for module that is not on server yet -->
        <module src="C:/Workspace/newmodule/bin-debug/newmodule.def"
            basePath="C:/Workspace/newmodule/bin-debug" />
    </modules>
</debug>
```

Root node should contain `server` and can contain `modules` nodes.

`server`
>   Sets the server domain name Pronto! should connect to. It can have the following properties:

>   `host`
>>      Server host

>   `port`
>>      Optional. Server port. Default is 8100

>   `secure`
>>      Optional. If you want to use secure connection, set it to `yes`, otherwise - to `no`. Default value is `no`.

`modules`
>   Optional. The list of local base paths for every module that should be loaded - both local and remote. Can contain any number of `module` tags, with the following attributes:

>   `id`
>>      Module id. If not specified, `src` should be specified. If specified, `src` can not be specified.

>   `src`
>>      .def file relative path. If not specified, `id` should be specified. If specified, `id` can not be specified.

>   `basePath`
>>      Relative path to the folder which contains module .swf, icons and other assets.

---

CommuniGate **Pro**

| **Pronto!** | **Module API** | **History** |
|---|---|---|

# Pronto! Revision History

- **6.2**
- **6.1**
- **6.0**
- **4.x**
- **3.x**
- **2.x**

**6.2.2** [CommuniGate Pro 6.2.2](#)

```
Pronto HTML
```

- General: improved the UI security .
- General: folder size is shown on mouse roll-over.
- General: the initial support for encrypted folders has been implemented.
- Dialer: the initial support for WebRTC in Safari has been implemented.
- Dialer: hot camera plug/unplug is supported now.
- Contacts: the `IMPP` vCard attribute with custom IM address is supported now.
- Mail: now multi-mailbox search can find FLAGGED and UNREAD messages.
- Dialer: fixed problem with the GIPS plugin installation in IE browser.
- Dialer, IM: fixed bug with chats and calls not showing peer status.
- Mail: fixed bug with the "Subscribe" button missing from mails with invitation to shared folders.
- Mail: it was not possible to actually rename the `INBOX` folder.
- Mail: fixed the bug when email attachments could not be downloaded as archive with Files module disabled.
- Mail: bug fix: inline images size grew significantly when they were included in HTML message replies.

**6.2.1** [CommuniGate Pro 6.2.1](#)

```
Pronto HTML
```

- General: sound customization preferences are implemented.
- General: hardened security of the interface against code injection attacks.
- Contacts: the presence state for items in Directory books is retrieved automatically.
- Mail: sorting preferences may be set individually for each folder now.
- Mail: the Junk flag set on messages is shown now.
- Mail: fixed the bug when images could not be inserted into plain text messages.

**6.2** [CommuniGate Pro 6.2.0](#)

```
Pronto Flash
```

- General: minor bugs have been fixed.

```
Pronto HTML
```

- General: minor bugs have been fixed.

**6.2c5** CommuniGate Pro 6.2c5

```
Pronto Flash
```

- General: fixed Problem with start of Air variant.

```
Pronto HTML
```

- General: fixed problems with modules dependencies.
- Mail: implemented defaults for "Notify When Delivered" and "Notify When Read" options.
- Mail: fixed bug with stuck Redirect window and multiple redirected copies.
- Mail: the selected sorting order is preserved now.
- Files: implemented preference for time to live for uploaded temp files.
- Files: fixed bug with incorrect folder structure reported in the inverted tree mode.
- Files: fixed the problem when a local file attached as a link multiple times resulted in broken links.
- Files: integration with OnlyOffice now use the language selected in account preferences.
- Contacts: fixed the bug when scrolling addresses list did not reveal new items.
- Calendar: fixed bugs with removing attachments in the Event preview panel.
- Calendar: fixed the bug with editing exceptions to recurring events.
- Calendar: fixed bug with removing an exception from a recurrent event: attendees were not notified.

**6.2c4** CommuniGate Pro 6.2c4

```
Pronto Flash
```

- General: `mailto:` links did not handle escaped sequences like `%XX`.
- General: links might be not escaped as necessary.
- Mail: inline images might be not shown in print preview.

```
Pronto HTML
```

- New module: the Notes module has been implemented.
- General: improved security of the interface.
- General: added function to delegate account rights.
- General: several UI and usability bugs were fixed.
- General: the "organization unit" display field has been added to auto-complete results.
- General: "hidden popups" window might be shown for some short time right after login.
- General: the option to auto-logout idle session has been implemented.
- General: the option to stay signed in has been implemented.
- Calendar: calendars were not shown to users with the "lookup" access right.
- Calendar: tentative items can be distinguished from items with other status.
- Calendar: implemented function to forward calendar events as invitations.
- Calendar: fixed bug with Free/Busy data not shown for some user names.
- Calendar: fixed bug with attaching files to events.
- Contacts: the Note field now allows for multiple lines.
- Contacts: added preferences for Directory-based books.
- Contacts: contact records can be sorted now.
- Contacts: dataset-based address books can be selected now.
- IM: permanent chat groups have owner now and can be removed by the owner.

- IM: buddies presence might not be shown.
- IM: implemented the option to send files from account storage to chats.
- IM: the special "mobile" presence status is supported now.
- Mail: content of decrypted mail messages might be not shown properly.
- Mail: the support for multiple identities has been improved.
- Mail: the favorite mailboxes list might cause operational problems with original mailboxes.
- Mail: now it's possible to edit documents in OnlyOffice directly from message attachments.
- Mail: the "Copy All Mail to" simplified rule has been updated.
- Mail: fixed the problem when sorting could be applied only to the current page.
- Mail: fixed the bug with "null" text inserted in the case of an empty html signature.
- Mail: fixed bug with incorrect behavior on the "send response" hot key.
- Mail: fixed bug when new messages were added into wrong positions in a sorted list.
- Mail: fixed bug when a wrong positions was selected after message deletion.
- Files: the "save as" feature for OnlyOffice integration has been added.
- Files: some operation on files and folders were not possible if their parent folder was renamed.
- Files: fixed wrong behavior while co-editing documents with OnlyOffice.
- Signals: fixed bug with incorrect initialization for receiving incoming calls.

**6.2c3** [CommuniGate Pro 6.2c3](#)

`Pronto HTML`

- Files: new faster method for large files uploading has been implemented.
- Mail: implemented option to recall messages silently.
- Mail: favorite folders are displayed in the folder list.
- Mail: context menus for message list management have been implemented.
- Contacts: the set of display fields for directory-based address books has been extended.
- Calendar: now it's possible to subscribe to external web calendars.

**6.2c2** [CommuniGate Pro 6.2c2](#)

`Pronto HTML`

- General: fixed problem with login after failed login using URL parameters.
- IM: Added support permanent chat groups.
- IM: Implemented list of recent chats.
- IM: Implemented interface for custom roster groups.
- IM: Improved sorting of active chats for better usability.
- Contacts: fixed the problem with contact record duplication.
- Contacts: implemented option to request presence info on contacts that are not in the roster.
- Contacts: searching might fail.
- Contacts: for some roster items subscription request might be not shown.
- Contacts: public information on just added buddy might be not retrieved properly.
- Mail: added option to confirm sending of a message without a subject.
- Mail: implemented support for composing of reply messages in the HTML format.
- Mail: implemented support for signatures in HTML format.
- Mail: implemented the function to recall recently sent messages.
- Mail: the Favourites panel to access most used folders has been implemented.
- Mail: fixed exception while showing some HTML-formatted messages.
- Mail: composer might be not shown in the list view mode.
- Mail: multi-message operations have been improved.
- Mail: read receipts are properly generated now.

**6.2c1** [CommuniGate Pro 6.2c1](#)

- General: Added support for the two-factor authentication.
- General: Added support for the forced password change.

| Pronto! | Module API | History |
|---|---|---|

# Pronto! Revision History

- 6.2
- 6.1.19
- 6.0
- 4.x
- 3.x
- 2.x

**6.1.19** [CommuniGate Pro 6.1.19](CommuniGate Pro 6.1.19)

```
Pronto HTML
```

- General: minor fixes.
- General: hardened protections agains code injection attacks.

**6.1.18** [CommuniGate Pro 6.1.18](CommuniGate Pro 6.1.18)

```
Pronto HTML
```

- General: implemented support for restoring abnormally closed resent sessions.
- Mail: content of mail messages might be not shown properly in IE browsers.
- IM: fixed bug when links sent in chats could not be opened.
- IM: fixed bug that prevented correct display of chat history.
- Preferences: fixed problem drop-down menus in extended Mail processing rules.

**6.1.17** [CommuniGate Pro 6.1.17](CommuniGate Pro 6.1.17)

```
Pronto HTML
```

- General: security vulnerabilities have been identified and closed.
- General: UI and usability bugs have been fixed.
- General: the "organization unit" display field has been added to auto-complete results.
- Mail: content of decrypted mail messages might be not shown properly.
- Mail: fixed bug when new messages were added into wrong positions to the sorted list.
- Mail: fixed bug with deleting items from multi-mailbox search results.
- Mail: fixed bug when sorting was applied only to visible page.
- Calendar: calendars were not shown to users with the "lookup" access right.

- Calendar: implemented option to forward calendar event as an invitation.
- Contacts: fixed bug with the Notes field in contact records showing only one line.
- Dialer: fixed problem when calls could not be started without camera attached.

### 6.1.16 CommuniGate Pro 6.1.16

`Pronto HTML`

- General: fixed problem with login after failed login using URL parameters.
- Contacts; contact records might be replaced with Buddies.
- Contacts: searching might fail.
- Contacts: for some roster items subscription request might be not shown.
- Contacts: public information on just added buddie might be not retrieved properly.
- Mail: fixed exception while showing some HTML-formatted messages.
- Mail: composer might be not shown in the list view mode.
- Mail: multi-message operations have been improved.
- Mail: read receipts are properly generated now.

### 6.1.15 CommuniGate Pro 6.1.15

`Pronto HTML`

- General: some minor bugs were fixed.
- Preferences: the panel showing active sessions has been added.
- Files: the integration with OnlyOffice has been improved; documents are opened within the Pronto interface.
- Files: the compatibility of file locking function with WebDAV clients has been improved.

### 6.1.14 CommuniGate Pro 6.1.14

`Pronto HTML`

- General: shortcut keyboard combinations have been optimized for different OSes.
- General: the help panel listing all shortcut keyboard combinations has been added.
- General: account presence is correctly updated now, in accordance with other registered devices.
- Mail: the Print operations did not work for messages in nested sub-folders.
- Mail: encrypted message content might be not displayed after unlocking S/MIME key.
- IM: added preference item to control delivery of IM to all devices.

### 6.1.13 CommuniGate Pro 6.1.13

`Pronto HTML`

- Mail: improved stability and productivity with long message lists.
- Mail: added functionality to remove messages found in multiple folders.
- Mail: message selection is not changed when a filter is applied.
- Mail: improved usability with keyboard shortcuts.
- IM: now messages are sent to all registered devices (not just the last one used in a dialog).
- IM: added option to delete all chats history.
- IM: added reload history feature, to get new messages typed on other devices.
- Files: improved integration with OnlyOffice: "co-editting" and "open as" functions are available now.
- Dialer: added dial-tones.
- Dialer: added option to delete all calls history.
- Mobile version: user interface usability has been improved.

### 6.1.12 CommuniGate Pro 6.1.12

`Pronto HTML`

- Mobile browsers: added support for the Calendar and Files modules.
- General: added function to show hidden pop-ups.
- Mail: added multi-composing functionality.
- Mail: implement page-based navigation.
- Dialer: added keypad to call pop-up.
- Dialer: added feature to transfer current call to other registered devices.
- Calendar: added the list view mode.
- IM: added search in chat logs feature.

```
Pronto Flash
```

- General: minor UI and stability fixes.

## 6.1.11 [CommuniGate Pro 6.1.11](#)

```
Pronto HTML
```

- General: Responsive design for tablets is improved.
- General: UI and stability fixes.
- Mail: the option to attach files ffrom account file storage.
- Mail: the option to store attachments into account file storage.
- Mail: the function to download attachments as ZIP archives.
- Mail: the option to send attachments as links to account's file storage.
- Dialer: full screen mode for video calls is implemented.
- Dialer: added video call option to call pop-up.
- Dialer: the "mute" option is implemented for the call notification panel.
- Chat, Mail: temporary files delivered in chat sessions or as links in mails are removed automatically now.
- Chat: opened chats can be re-opened after re-login.
- Calendar: added working days selection to the preferences panel.
- Files: implemented option for the straight view for file storage hierarchy.

```
Pronto Flash
```

- General: minor UI and stability fixes.

## 6.1.10 [CommuniGate Pro 6.1.10](#)

```
Pronto HTML
```

- General: Responsive design is added.
- General: Improved detection of old browsers (like IE 7,8) for info on unsupported browsers on the login page.
- General: the "Fresh" skin has been updated.
- General: many issues in UI has been fixed, added missing translations.
- Mail: new option to add e-mails as attachments.
- Mail: shortcuts for the most common operations have been added.
- Mail: the Vacation rule functionality has been extended.
- Mail: the "vertical" layout has been implemented.
- Dialer: calls can be controlled in pop-up windows now.
- Files: automatic file locking an unlocking of subscribed files during upload/download operations has been implemented.
- Files: special folder names can be shown translated in UI.
- Files: the feature to switch the root of files tree view between 'private' and 'public'.
- Files: the option to hide file transfer function in chats has been added.

```
Pronto Flash
```

- Mail: the Vacation rule functionality has been extended.

**6.1.9** [CommuniGate Pro 6.1.9](#)

`Pronto HTML`

- General: the "fresh" skin is used by default now.
- General: the warning on page reloads is implemented (off by default).
- General: the support for "mailto:" links has been improved.
- Preferences: password panel appearance has been improved.
- Files: file locking functionality is implemented integrated with the OnlyOffice service.
- Files: sending files in chats has been improved.
- Calendar: the "Free/Busy Publishing" preference has been added.
- Mail: the option to show raw message source has been added.
- Mail: the message redirection function has been implemented.
- Bug Fix: Contacts: dataset records were not updated on filtering operations.
- Contacts: now Directory records are listed in contact selector.
- Dialer: the option to make calls via all devices has been implemented.

`Pronto Flash`

- Bug Fix: Mail: year could change incorrectly in the week view.

**6.1.8** [CommuniGate Pro 6.1.8](#)

`Pronto HTML`

- Files: now it is possible to set ACL for a folder folder with all its sub-folders.
- Integration with OnlyOffice has been implemented.
- Dialer: video calls with webRTC are now enabled.
- Dialer: new preference to switch the view on call start.
- Mail: new preference to show number of messages in mailboxes.
- Chats: chats are now using separate pop up windows.
- Contacts: implemented the pop up window with buddy list.
- Bug Fix: Files: issues with file ACLs fixed.

`Pronto Flash`

- Bug Fix: Mail: dragging the "INBOX" folder was not correctly processed.
- Contacts: the "userCertificate" field removed from the directory search request.

**6.1.7** [CommuniGate Pro 6.1.7](#)

`Pronto HTML`

- Core: improved processing for hidden modules.
- Dialer: added notification about active call to browser UI.
- Dialer: now it's possible to see the contact card during the call.
- Dialer: added support for new WebRTC media transport specifications.
- Dialer: improved UI for call transfer operations.
- Bug Fix: Mail: message flags might be not set properly.
- Bug Fix: Mail: subfolders in the folder list might be sorted incorrectly.

**6.1.6** [CommuniGate Pro 6.1.6](#)

`Pronto HTML`

Calendar: added free/busy panel for event attendees.

- Dialer: notifications for missed calls from teh same caller are grouped now.
- Dialer: added preferences for automatic call accepting.

```
Pronto Flash
```

- Bug Fix: Address Book: recipient lists formed from multiple groups were not handled properly.

**6.1.5** [CommuniGate Pro 6.1.5](#)

```
Pronto HTML
```

- Bug Fix: General: fixed problems introduced with addition of the new skin.
- Bug Fix: Chats: smilies were not displayed in FireFox.
- Bug Fix: Preferences: fixed UI controls behaviour.
- Bug Fix: a skin might be used by a user even when not allowed by domain administrator.

```
Pronto Flash
```

- Address Book: Improved address record conversion from Directory-Based to Regular address books.
- Bug Fix: Address Book: attributes with multiple values were shown incorrectly in Directory-based address books.

**6.1.4** [CommuniGate Pro 6.1.4](#)

```
Pronto HTML
```

- Core: Minor fixes in UI.
- Bug Fix: Mail: Fixed memory leaks that caused slow rendering of messages.
- Bug Fix: General: removed old debugging code for anonymous statistics usage collection.
- Bug Fix: General: the Compose button might be not available in the Classic skin variant.

```
Pronto Flash
```

- Core: Minor fixes in UI.
- Bug Fix: General: removed old debugging code for anonymous statistics usage collection.

**6.1.3** [CommuniGate Pro 6.1.3](#)

```
Pronto HTML
```

- General: A new skin "Fresh" is now available. Skins can be changed in preferences.
- Bug Fix: Mail: Users were allowed to edit mail rules with prohibited actions.
- Bug Fix: Contacts: Retrieving records from directory-based books has been optimized.
- Bug Fix: IM: Group chats could appear broken in Firefox.
- Bug Fix: Voice: Redirect option for incoming calls might not work in some cases.
- Bug Fix: Voice: Display of the call in progress was broken in Internet Explorer 11.
  .
- Bug Fix, Voice: Clicking on incoming call system notification could decline the call in Firefox.
- Core: Minor fixes in UI.

```
Pronto Flash
```

- Mail: Messages can now be marked as "read" when "Filter Unread" is enabled.
- Mail: The Microsoft-specific "winmail.dat" attachments can now be displayed and downloaded.
- Bug Fix: Mail: Deleting of an attachment could delete the text in the message body.
- Bug Fix: Mail: Signature was not inserted into a forwarded message when forward setting is "inline".
- Bug Fix: Mail: Users were allowed to edit mail rules with prohibited actions.

- Bug Fix: Calendar: Every participant received an "event cancelled" message if one of the participants deleted the accepted event.

**6.1.2** CommuniGate Pro 6.1.2

```
Pronto HTML
```

- Contacts: Directory-type contact lists are fully supported now.
- Core: Minor bug fixes.

```
Pronto Flash
```

- Core: address auto-complete function has been improved.
- Dialer: fixed bug with microphone gain control (GIPS plugin).
- Core: the selection in the list of available spell-checkers defaults to Account's language.
- Core: mechanisms of communications to the external CG/PL scripts have been optimized.

**6.1** CommuniGate Pro 6.1

```
Pronto HTML
```

- Tasks: the Tasks module added.
- Preferences: The text of the vacation message is now preserved when the message is disabled.
- Contacts: Domain type contacts display added.
- Core: Closing session on browser close is implemented.
- Core: A number of performance and client-server communication optimizations have been implemented.
- Core: Faster loading of application achieved by library caching and module loading optimization.
- Core: Support for killOldSession URL parameter is added.

**6.1c1** CommuniGate Pro 6.1c1

```
Pronto HTML
```

- Dialer: support for new XIMSS call leg interface is implemented.
- Dialer: WebRTC support is implemented.
- EMail: the method to display addresses in message headers is improved.
- Email: one-column view is implemented.
- Preferences: when the vacation message is switched off, its text is preserved.

---

| Pronto! | Module API | History |
|---------|-----------|---------|

# Pronto! Revision History

- **6.1**
- **6.0.10**
- **6.0.9**
- **6.0.8**
- **6.0.7**
- **6.0.6**
- **6.0.5**
- **6.0.3**
- **6.0.1**
- **6.0**
- **4.x**
- **3.x**
- **2.x**

**6.0.10** [CommuniGate Pro 6.0.10](#)

```
Pronto HTML
```

- General: the functionality split into modules to improve load times and to allow feature management.
- General: many minor UI imprevements.
- General: the logoutURL GET paramater support is implemented.
- General: LogoutJumpRef string parameter is supporedt.
- General: an optional Logout button is implemented.
- General: auto-reconnect functionality is improved.
- General: Intenet Explorer 9 is no longer supported.
- Mail: Mail Rule management is implemneted.
- Mail: Mailbox access rights management is implemented.
- Mail: Mailbox Aliase management is implemented.
- Mail: the Folder Edit UI is updated.
- Mail: Vacation Message and Redirect All Rule management is implemented.
- Mail: Compact header view is improved.
- Mail: the Select All function is implemented.
- Mail: S/MIME can be disabled now.
- Mail: Report Spam button is implemented.
- Mail: Drag-and-drop is implemented for addresses in the Compose window.
- Mail: the Contact actions menu is implemented.
- Calendar: Public web calendar sharing is implemented.
- Calendar: the View Event popup is implemented.
- Calendar: Import/Export functionality is implemented.
- Contacts: support for Datasets is implemented.

- Contacts: support for Contact Groups is implemented.
- Contacts: "Yomi" names are displayed now.
- IM: an IM session can be started with an arbitrary contact now.
- Files: the Refresh button is added.
- Dialer: the Preferences panel is implemented.
- Dialer: Server-side conferences support is implemented.
- Dialer: Simple Rules are implemented.
- Dialer: Preferences: the "Install Plugin" button is implemented.
- Dialer: Video Calling is disabled now due to changes in WebRTC implementation.
- Bug Fix: General/Preferences: Mute Sound didn't mute all the sounds.
- Bug Fix: Mail: several scrollbar problems are fixed.
- Bug Fix: Mail: right-click did not work to paste addresses into the Compose dialog.
- Bug Fix: Calendar: multi-day events were created incorrectly.
- Bug Fix: IM: chatroom selection window was never closed.
- Bug Fix: IM: URL was not recognized in IM messages in some cases.
- Bug Fix: IM: the incoming message sound was played for own messages in group chats.
- Bug Fix: Files: current folder click did not work in the breadcrumbs.
- Bug Fix: Files: shared folder link was formed incorrectly.
- Bug Fix: Files: folder subscription was not displayed in some cases.

```
Pronto Flash
```

- Mail: Email addresses are checked before messages are sent.
- Mail: Desktop: Attachments are opened usiging the default OS application.
- IM: A preference is added to add Buddies to the roster automatically.
- Dialer: the mute-all/unmute-all feature (available to the conference organizer) is implemented.
- Dialer: the Push-to-talk functionality is implemented.
- Dialer: the Compact view for the audio conference participant list is implemented.
- General: Multiple instances of Pronto can be installed in the same Domain, using different Skins.
- Bug Fix: Mail: Message receive time was changed when its attachments were deleted automatically.
- Bug Fix: Dialer: DTMF could be sent incorrectly.
- Bug Fix: General: Offline alerts were not displayed after login.

**6.0.9** [CommuniGate Pro 6.0.9](#)

- Pronto Lite: General: improved support for Internet explorer.
- Pronto Lite: General: notifications use modern Web Notifications API now.
- Pronto Lite: Mail: the support for Multiple Identities is implemented.
- Pronto Lite: Mail: S/MIME preferences added.
- Pronto Lite: Mail: Inline reply actions added for forwarded messages.
- Pronto Lite: Mail: INBOX can be renamed now.
- Pronto Lite: Mail: periodic clean up for the Trash folder is implemented.
- Pronto Lite: Mail: rules management is implemented.
- Pronto Lite: Mail: recipients are added to the address book on reply now.
- Pronto Lite: Calendar: Proper time is selected in dropdown list when creating new event now.
- Pronto Lite: Calendar: Access to publicly shared calendars is implemented.
- Pronto Lite: Contacts: performance improvements when browsing contacts folders.
- Pronto Lite: IM: various UI improvements in group chats: avatars and real names are displayed now.
- Pronto Lite: Files: minor UI improvements.

- Pronto Lite: Bug fix: Mail: Messages in Japanese(ISO) encoding were not sent.
- Pronto Lite: Bug fix: Mail: 'Forward' flag wasn't set when a message was forwarded inline.
- Pronto Lite: Bug fix: Mail: 'No recipient specified' note was displayed when forwarding message to a new recipient.
- Pronto Lite: Bug fix: Mail: Text starting with < symbol could not be pasted.
- Pronto Lite: Bug fix: Mail: Incoming mail notifications didn't work until explicitly turned on.
- Pronto Lite: Bug fix: Dialer: conference participants were not displayed in certain call configurations.
- Pronto Lite: Bug fix: Calendar: Event attachments were lost after performing drag-n-drop to another time.
- Pronto Lite: Bug fix: Login: layout fixes for modern WebKit browsers.
-
- Mail: Implemented setting to control the availability ov the Multiple identities feature.
- Mail: URL to the script to verify identities can be customized now.
- Mail: periodic clean up for the Trash folder is implemented.
- Presentation: converter now works with jre7, the Presentation module loads faster.
- Mail: the focus is kept on previously selected message afetr order/sorting change.
- Bug fix: Mail: Some specifically formatted messages might be displayed incorrectly.
- Bug fix: Mail: The proper message flag was not set on messages forwarded inline.

**6.0.8** [CommuniGate Pro 6.0.8](#)

- Pronto Lite: Files: the module is implemented.
- Pronto Lite: General: IE11 compatibility.
- Pronto Lite: General: the interface language can be changed by user now.
- Pronto Lite: General: users can change passwords now.
- Pronto Lite: Mail/Calendar/Contacts: Mailbox ACL and sharing are implemented.
- Pronto Lite: Mail: auto-complete works for the default address book now.
- Pronto Lite: Mail: `mailto:` links open the Compose dialog now.
- Pronto Lite: Mail: Editor: images can be copy-and-pasted into the editor now.
- Pronto Lite: Mail: the Preferences panel is implemented.
- Pronto Lite: Mail: columns can be resized now.
- Pronto Lite: Mail: printing function is implemented.
- Pronto Lite: Mail: Shift+Click can be used for range selections now.
- Pronto Lite: Mail: the Empty Trash function is implemented.
- Pronto Lite: Calendar: the Preferences panel is implemented.
- Pronto Lite: Contacts: the support for contact groups is implemented.
- Pronto Lite: Contacts: the import/export vCards functions are implemented.
- Pronto Lite: IM: Public Chat rooms are supported now.
- Pronto Lite: Dialer: calls can be merged into a conference now.
- Pronto Lite: Dialer: adding participants to a conference has been reimplemented.
- Pronto Lite: General: lots of smaller improvements across all modules.
-
- Mail: the support for Multiple Identities is implemented.
- Dialer: plugin installation compatibility for Firefox 25 and newer.
- Mail/Dialer: rules priorites can be set manually now.
- Bug fix: Files: shared files links could include wrong port number.
- Bug fix: Calendar: Requests were sent to event subscribers even if "Send requests" was unchecked.
- Bug fix: Mail: Some columns were not properly sycnhronized between WebUser and Pronto.
- Bug fix: Mail: inline images were not printed.
- Bug fix: Calendar: month switching through small calendar failed in certain cases.

**6.0.7** [CommuniGate Pro 6.0.7](#)

- Mail: the support for the `X-Color` header is implemented.
- Mail: the support for quoted local part in email addresses is implemented.

**6.0.6** [CommuniGate Pro 6.0.6](#)

- Pronto Lite: rewritten from scratch with modern HTML5 technologies allowing better load times, customization and extensibility.
- Pronto Lite: the Mail module has been implemented.
- Pronto Lite: the Calendar module has been implemented.
- Pronto Lite: the Dialer module supports WebRTC implementation in Chrome.
- Mail: the `Sender` header is displayed if the `From` and `Sender` header values do not match
- Mail: the "Select all" action button is added to the address confirmation dialog (selects local addresses).
- Mail: the "Confirm recipients" dialog is added.
- Mail: the preference option to always show plain text is implemented.
- Mail: the dialog layout has been modified to better accommodate messages in Japanese.
- Mail: Junk messages are now marked with a different color in the message grid.
- Mail: User can explicitly decline sending confirm receipt now.
- Dialer: plugin is installed in Internet Explorer 10 now.
- Bug fix: Contacts: Yomi fields were not displayed if a contact was been created via Outlook.
- Bug fix: Contacts: corrected formatting of Japanese names (removed comma).
- Bug fix: Mail: the Change font preference worked incorrectly for text/plain letters in Japanese.

**6.0.5** [CommuniGate Pro 6.0.5](#)

- Mail/Calendar: error is displayed now if target event is not found for calendar status update.
- Mail/Contacts: vCards can be emailed now.
- Dialer: media plugin installation works in Internet Explorer 10 now.
- Pronto Android: Plugin version 2.0.12.0 is used.
- Pronto Android: Now suggested as a method (along with other VoIP clients) for outgoing calls.
- Bug fix: Mail: HTML encoding was not properly specified for ISO-2022-JP.
- Bug fix: Dialer: Video window may be shown when the call is placed on hold.
- Bug fix: Mail: Take Certificate duplicated the contact.
- Bug fix: Mail: Take Certificate might not work if there was no corresponding contact in the address book.
- Bug fix: Mail: vCard attachments were not properly saved to local and server storage.
- Bug fix: Pronto Lite: IM notifications did not work in Firefox and Internet Explorer.
- Bug fix: Pronto Android: fixed the Description field (multiline is allowed) in the CrashLogReport.
- Bug fix: Pronto Android: Mute did not work after Hold.

**6.0.3** [CommuniGate Pro 6.0.3](#)

- Dialer: All Calls/Missed/Incoming/Outgoing selector added.
- Bug fix: Mail: Pronto could freeze when replying to certain messages.

**6.0.1** [CommuniGate Pro 6.0.1](#)

- Mail: now Contact Groups are included into auto-complete suggestions.
- Calendar: event reminder localization is improved.
- Mail: S/MIME: certificate time data format is localized now.
- Bug Fix: Mail: S/MIME certificate creation did not work.

**6.0** [CommuniGate Pro 6.0](#)

- Bug Fix: Calendar: December 2012 could be skipped in monthly navigation.
- Bug Fix: Calendar: The end of recurrence could be saved incorrectly.
- Bug Fix: Calendar: Notification about recurring All Day event could be displayed multiple times.

Bug Fix: E-mail: Notes: Autosaving could work incorrectly in some cases.

- Bug Fix: E-mail: Multiple attachments could be downloaded incorrectly.
- Bug Fix: Contacts: Group containing contacts in a certain format could be handled incorrectly.
- Bug Fix: Contacts: Drag and Drop of a contact to a group could close the group edit panel.

# CommuniGate Pro

| Pronto! | Module API | History |
|---------|-----------|---------|

# Pronto! Revision History

- **6.0**
- **4.10**
- **4.9**
- **4.7**
- **4.6**
- **4.5**
- **4.4**
- **4.3**
- **4.2**
- **4.1**
- **4.0**
- **3.x**
- **2.x**

**4.10** CommuniGate Pro 5.4.10

- Dialer: Device list is refreshed when Preferences window is opened.
- Files: Preferred URL scheme and port for file sharing can be customized now through strings.data
- Login: PLAIN login method is allowed now for HTTP connection.
- Mail: Files from subscribed folder can be attached to messages now.
- Mail: 'Sent' header is displayed instead of 'Received' header when forwarding message inline.
- Mail: Timezone is dispayed in 'Sent' header when forwarding message inline.
- Bug fix: Calendar: Event with weekly recurrence couldn't be saved.
- Bug fix: Customization: skin icons were not loaded in some cases.
- Bug fix: Dialer: Hold and Mute didn't work for video calls.
- Bug fix: Dialer: Secure call icon didn't appear in some cases when call was actually secure.
- Bug fix: Files: File sharing notification letter didn't specify proper encoding.
- Bug fix: IM: messages were not marked as read when all-chat window was opened from taskbar.
- Bug fix: Notes: Note created in Pronto 4 can't be opened in WebUser interface.
- Bug fix: Notes: Note lost focus during auto-save.
- Bug fix: Notes: Long subject wasn't properly displayed.
- Bug fix: Notes: Notes created in Outlook were not properly displayed and saved when edited.
- Bug fix: Presentations: long file names were not handled correctly.
- Bug fix: Shell: 'Dismiss all' removed incoming call notification.

**4.9** CommuniGate Pro 5.4.9

- IM: Secure file sending has been implemented.
- Preferences: 'Replied addresses' are now deleted when the Vacation Message option is disabled.
- SMS: formatting symbols are removed from the SMS phone number.
- Whiteboard: Module is listed in the navigation panel.
- AIR: E-mail: Attachments can be dragged into a system folder.
- AIR: Files: Files can be copied/moved by drag and drop.
- Dialer: Plugin installation instructions have been adjusted for the Chrome Web Store policy.
- E-mail: E-mails can be attached to a message by drag and drop into the message compose window.
- Tasks: the description field is enlarged.
- Bug Fix: Core: HTTP Binding connections could lose server responses under poor network conditions.
- Bug Fix: Login: the Logo could be mirrored for RTL UI.
- Bug Fix: AIR: cache files were never cleaned up.
- Bug Fix: E-mail: an inline image could stay as an attachment after it being deleted from the message body.
- Bug Fix: E-mail: the "Empty trash" button was not localized.
- Bug Fix: E-mail: the "Accept.." button sometime did not work on the 1st click.
- Bug Fix: E-mail: the "Show HTML" link was displayed for messages without any HTML part.
- Bug Fix: Contacts: a japanese text could overlay the grey hint inside the edit dialog fields.
- Bug Fix: Calendar: recurring events could be displayed twice.
- Bug Fix: Dialer: the Call button did not open the minimized call dialog.
- Bug Fix: Tasks: the shared tasks functionality could work incorrectly.
- Bug Fix: Notes: the selected note was not displayed in preview after message sorting.
- Bug Fix: Whiteboard: adding a new participant could throw an exception.
- Bug Fix: IM: some emoticons were not recognized.
- Bug Fix: IM: declining a buddy invitation added the contact to "Banned" dataset.
- Bug Fix: IM: the chat room header context menu could throw an exception.
- Bug Fix: IM: the chat room header context menu could contain incorrect items.
- Bug Fix: IM: the new message counter could be incorrect.
- Bug Fix: IM: notifications for group chats did not contain the chat name.
- Bug Fix: IM: the "Show Public info" operation did not request the current user profile.
- Bug Fix: IM: Text could become unselectable after window mode switching.
- Bug Fix: Photos: the file-tree pane could not be scrolled in some cases.
- Bug Fix: Preferences: E-mail addresses in rules were not saved in some cases.

**4.7** CommuniGate Pro 5.4.7

- Core: Modules can be now configured as Disabled (not loading), Visible (with an icon on the toolbar) and Hidden (under "More" button).
- Core: Sound framework and preferences have been implemented.
- E-mail, Calendar, Contacts: visual indication for shared folders is implemented.
- E-mail, Calendar, Contacts: folder sharing E-mail notifications are implemented.
- Files: AIR: Possibility to drag files from the files module directly to the desktop is implemented.
- E-mail: Multiple files can be selected as attachments in the server file selection dialog.
- IM: XMPP ping support has been added.
- IM: nick names for chat rooms can be selected now.
- Notes: Rich text editor has been implemented.
- Calendar: Events in shared folders with full access rights are now editable.
- Calendar: the "Access granted to" dialog can work with sub-folders now.
- Dialer: A new preference added for the server conference "Add me" functionality to work as an incoming call, which the user can accept on any device.
- Bug Fix: Calendar: Calendar functionality could be available even if the WebCal service was disabled.
- Bug Fix: Calendar: Editing of an event created via iCal could throw an error.
- Bug Fix: Calendar: Move or Copy events dialog incorrectly displayed calendar names.
- Bug Fix: E-mail: A "+" button in server file selection dialog could freeze the UI.

Bug Fix: E-mail: the "Show undecoded message" link for a forwarded message could open the original message instead.

- Bug Fix: E-mail: the "Show HTML" link in the message header could open an incorrect message.
- Bug Fix: E-mail: [x] button in Signed / Encrypted window could send the message.
- Bug Fix: E-mail: the "Certificate problem" message changed to the proper one.
- Bug Fix: Contacts: AIR: Contact view dialog could open blank in some cases.
- Bug Fix: General: A requirement for the HTTP service was excessive in some modules.
- Bug Fix: Music: Sharing link was created incorrectly.
- Bug Fix: Twitter: Wall was not refreshed after sending a post.
- Bug Fix: Twitter: Not all the options for a new post were displayed.
- Bug Fix: Dialer: the "Use STUN" preference was "On" by default.
- Bug Fix: Dialer: the 'Join to' option did not work in Mac OS.
- Bug Fix: Dialer: the 'Unhold' button did not work in Mac OS.
- Bug Fix: Dialer: Preferences were not always applied after switching an account.
- Bug Fix: Dialer: Phone numbers containing a '#' could be resolved incorrectly.
- Bug Fix: Dialer: The incoming call alert could not display the contact name.
- Bug Fix: IM: chat display was broken in some situations.

## 4.6 CommuniGate Pro 5.4.6

- E-mail: attaching the profile vCard to an E-mail message is implemented.
- E-mail: vCard attachments are recognized and processed.
- E-mail: the 'MIME-Version: 1.0' header field is added to all messages now.
- E-mail: full HTML headers are included into HTML message parts.
- E-mail: server-stored files can be attached to E-mail messages now.
- E-mail: a new preference to disable the "RecentAddresses" dataset has been added.
- Core: the "Dismiss all" notifications operation is implemented.
- AIR: Login: the secure mode selector can be defined using the strings file.
- AIR: Login: the Secure connection mode can be specified using the strings file.
- Login: redirection to a custom signup page is implemented.
- Login: language selection dialog is implemented.
- Bug Fix: Calendar: Web Calendars did not display events in some cases.
- Bug Fix: Calendar: All Day events could be saved with incorrect end times.
- Bug Fix: Calendar: An invitation for a recurrent event was displayed with incorrect time if the organizer used a different time zone.
- Bug Fix: Calendar: the OFFICE vCard attribute was used instead of the WORK attribute.
- Bug Fix: E-mail: Message body could be displayed as empty when a draft letter was opened.
- Bug Fix: AIR: the "Enter" key could become disabled during relogin.
- Bug Fix: AIR: the Host input field did not work correctly.
- Bug Fix: IM: Offline message notification displayed an incorrect time stamp.
- Bug Fix: IM: File downloading from IM chat did not work in some cases.
- Bug Fix: Gallery: the "Slide show" mode could work incorrectly.
- Bug Fix: Gallery: IE: An exception could be thrown when on album opening.

## 4.5 CommuniGate Pro 5.4.5

- E-mail: a new preference to request Read confirmation for all outgoing messages.
- E-mail:the "Show HTML" button is displayed for some messages.
- General: added a link to install Pronto! Pro from the Pronto! Web login page.
- Twitter: public timeline can be hidden now.
- Twitter: custom timeline can now be added as a default.
- Video chat: one-to-one sessions are implemented.
- IM: ClientID is now used for the IM communication.
- Calendar: Exception display is improved.

- S/MIME: UI is improved.
- Look and Feel: a separate skin for taskbar buttons has been created.
- Bug Fix: E-mail: move or copy operation could fail.
- Bug Fix: E-mail: Preferences: the column order could be incorrect.
- Bug Fix: E-mail: some HTML tags were processed incorrectly.
- Bug Fix: E-mail: opening a message forwarded as an attachment could result in an error.
- Bug Fix: Calendar: AIR: Drag-n-drop events did not work in some cases.
- Bug Fix: AIR: some buttons could be disabled when a user was removed from a conference call.
- Bug Fix: Skins: AdditionalProntoButtonNames parameter did not work.
- Bug Fix: Tasks: a new task could not be created in some cases.
- Bug Fix: Voice: simple signal rules were created with wrong priorities.
- Bug Fix: Music: Some labels were not localized.
- Bug Fix: Localization: Social and Notes module names in the task bar were not localized.

## 4.4 CommuniGate Pro 5.4.4

- E-mail: HTML lists are supported now.
- E-mail: added new preference to forward messages inline.
- SMS: messages can be sent to multiple destinations now.
- Voice: now the Attended Transfer is the primary transfer function if two or more calls are active.
- Voice: dragging the call with Ctrl initiates an attended transfer now.
- News: RSS source link modification is implemented.
- News: the UI is improved.
- General: tooltips added for the navigation bar module icons.
- Bug Fix: SMS: the leading "+" in destination addresses was cut off.
- Bug Fix: S/MIME: password modifications did not work.
- Bug Fix: E-mail: portions of HTML content could be duplicated.
- Bug Fix: E-mail: a compose window could be closed without a confirmation dialog.
- Bug Fix: E-mail: the MDN was sent for messages in Sent/Draft mailboxes.
- Bug Fix: E-mail: the Junk Mail preferences could function incorrectly.
- Bug Fix: Core: small images were streched when used as a wallpaper.
- Bug Fix: Core: the Password Modification panel was displayed when password modifications were not allowed.
- Bug Fix: Core: AIR: the Feedback button position was incorrect.
- Bug Fix: News: some context menu items did not work.
- Bug Fix: News: some feed content was not displayed.
- Bug Fix: News: subject lines could be cut short for some feeds.
- Bug fix: Files: file information polling was too aggressive.
- Bug fix: Voice: Call transfer errors were not always handled.
- Bug Fix: Voice: "On hold" status instead of "connected" could be shown after a tranfer.
- Bug Fix: Voice: addresses containing "#" could be processed incorrectly.

## 4.3 CommuniGate Pro 5.4.3

- RSS: the News module is implemented.
- E-mail: S/MIME support is implemented.
- Calendar: iCalendar export is implemented.
- Calendar: event Copy/Move operations are implemented.
- Voice: Call Info panel is implemented.
- Voice: Call Info is recorded in the Log.
- Core: the System menu has been moved from the "Logout" to the "Preferences" button.
- Bug Fix: Calendar: the Week Start day could be displayed incorrectly.
- Bug Fix: Calendar: Events were not displayed for some calendars.
- Bug Fix: Calendar: "Update attendee status" operation did not provide any visual feedback.
- Bug Fix: Contacts: A new address could not be added to an empty dataset.

- Bug Fix: E-mail: A recipient address was always pasted at the end of the "To:" field.
- Bug Fix: Contacts: The "Delete" key did not work for Dataset entries.
- Bug Fix: Contacts: The "Call" dropdown menu item in the preview pane could not be selected twice.
- Bug Fix: Contacts: No confirmation dialog was displayed when a modified Contact window was closing.
- Bug Fix: Contacts: The "Call" button in the preview toolbar was not displayed for the contacts containing only E-Mail addresses.
- Bug Fix: Contacts: The "E-mail" column could be duplicated in the Address Book grid.
- Bug Fix: AIR: MacOS: The cursor could disappear during Pronto initialization.
- Bug Fix: IM: Some buddies were displayed twice.
- Bug Fix: IM: MacOS: Tapping on the status did not display the drop down list.
- Bug Fix: IM: Chat window scrolling could work incorrectly.
- Bug Fix: IM: Messages could be displayed out of order.
- Bug Fix: IM: Links loaded from the IM history were displayed with duplicated slashes.
- Bug Fix: Videos: The slider could freeze when playing a video.

**4.2** CommuniGate Pro 5.4.2

- Voice: STUN/ICE support is implemented.
- Music: On-line radio is implemented.
- IM: Presence changing Alerts are implemented.
- Video: YouTube integration is implemented.
- Social: the Refresh button is added.
- Social: the latest Facebook API changes are supported.
- Notes: clickable links are supported now.
- Core: the "Use proxy-safe mode" Login option is now hidden if the server specifies HTTP Binding as the recommended connection method.
- IM: Server-side history is now supported.
- E-mail: line breaks are inserted into HTML message bodies to avoid very long lines.
- E-mail: the Call option is added to the contact context menu.
- Calendar: now only the time info is shown for foreign private events.
- Calendar: search functionality is implemented.
- Presentations: the Module is implemented.
- AIR: non-default ports are supported now.
- Login: Contact loading is accelerated.
- Bug Fix: Login: some Login parameters did not work.
- Bug Fix: Login: the Language URL Attribute was processed incorrectly.
- Bug Fix: E-mail: the Empty Trash button could not be disabled.
- Bug Fix: E-mail: the view was not refreshed switching the focus to some other window and back.
- Bug Fix: E-mail: Shift + Tab did not work for Mail Compose body field.
- Bug Fix: E-mail: AIR. Message body was not displayed after opening the E-Mail module for the 2nd time.
- Bug Fix: E-mail: some attachments could not be saved on the server.
- Bug Fix: E-mail: some HTML messages were displayed incorrectly.
- Bug Fix: E-mail: messages could remain striken out on slow internet connection.
- Bug Fix: E-mail: an error occured if the message body was larger than 250kb. Now it is displayed as an attachment.
- Bug Fix: Contacts: searching animation in auto-completion list did not stop in some cases.
- Bug Fix: Core: AIR: Pronto could not be loaded over some secure connections.
- Bug Fix: IM: Groups could be displayed incorrectly after dragging them to another place in the roster.
- Bug Fix: IM: an Avatar was not loaded when a new buddy was added to the roster.
- Bug Fix: IM: AIR: Notifications about incoming IMs remained in the Pronto header after the active chat had been closed.
- Bug Fix: IM: a File with a name containing special characters could not be opened when sent via IM.
- Bug Fix: Calendar: One occurance of a reccurent event could not be removed.
- Bug Fix: Calendar: Calendar was not refreshed when the Time Zone Preference had been changed.

- Bug Fix: Calendar: Pronto 4.x did not support calendar colours of Pronto 2.x.
- Bug Fix: Calendar: Program could freeze when switching calendar views.
- Bug Fix: Log: Log records with the same timestamp could be messed up.
- Bug Fix: Voice: CommuniGateAIRPluginGIPS was not always stopped after Pronto AIR client had been closed.
- Bug Fix: Tasks: a new task was accepted by default.
- Bug Fix: Tasks: the status was not updated immediately in some cases.
- Bug Fix: Tasks: a Task could not be removed after being marked as complete.
- Bug Fix: Tasks: attaching a file to a Task could cause an error.
- Bug Fix: Social: the Wall could stop displaying entries.
- Bug Fix: Social: a text could disappeared from input field after switching to some other window.
- Bug Fix: Social: image files with certain names could not be posted to Twitter.
- Bug Fix: Photos: the Error #2032 was thrown when opening photos with non-english names (IE only).
- Bug Fix: Photos: the Escape keystroke did not exit the Full-Screen mode.
- Bug Fix: Notes: 'E-Mail' and 'Delete' buttons did not work for attachments.
- Bug Fix: Notes: Text was not displayed correctly if an event-type (.isc) file was attached.
- Bug Fix: Music: Tracks with non-ASCII tag symbols could not be played.
- Bug Fix: Music: some genres were not displayed correctly.
- Bug Fix: Whiteboard: an absent user was shown in the attendees list.

**4.1** CommuniGate Pro 5.4.1

- Core: multiple display support is implemented.
- AIR: MS Office, .pdf, and other attachments are opened using the desktop applications.
- General: RSIP support has been added.
- General: pressing the Enter key in the Communicator phone number field starts a call.
- Files: user with Folder administrative rights can share files now.
- Files: New setting added that allows to set option 'hide/show system folders'.
- IM: custom status string is implemented.
- Phone: Signal Rules editor is implemented.
- Calendar: meeting spot search for a week period is implemented.
- Calendar: the Reminder option for accepted events is enabled now.
- Files: an option to create a new Storage folder when saving an attachment is implemented.
- E-mail: only a single copy of a message window is open now.
- E-mail: message headers can be scrolled now.
- Files: the empty path is shown instead of "/private" when private folders are being created.
- Files: the Text editor now opens files of an arbitrary size.
- Files: the status bars for storage size and number of files are displayed now.
- Files: Storage data is polled every 15 seconds now.
- Files: a text message for the unlimited storge size has been added.
- Files: now folder view refreshing refreshes all its subfolders, too.
- Whiteboard: Image resizing is implemented now.
- Bug Fix: General: Links with non-ASCII characters could not be open.
- Bug Fix: Core: SessionID-base logins did not work.
- Bug Fix: Core: the URL domain was not passed to the Server on login.
- Bug Fix: Core: non-ASCII characters in file names were not handled for .mp3 and .flv files.
- Bug Fix: AIR: a security problem when opening .swf files could result in a crash.
- Bug Fix: Calendar: the Monthly view was cut off when printing.
- Bug Fix: Calendar: an exception could be thrown on preferences opening.
- Bug Fix: Contacts: Contact preview was hanging if all E-mails from that contact have been deleted.
- Bug Fix: Contacts: It was not possible to select a folder for vCard import.
- Bug Fix: Contacts: Contacts with demaged avatars could not be opened.
- Bug Fix: Contacts: Some Contact fields could not be saved in when the Japanese or Chinese language was selected.
- Bug Fix: Files: Grid sorting did not work after module reopening.

- Bug Fix: E-mail: Mail dialogs were not formatted when replying to Pronto letters with Apple Mail.
- Bug Fix: E-mail: an exception was thrown when an inline image was being added.
- Bug Fix: E-mail: Simultaneous auto-save and send could result in exceptions.
- Bug Fix: E-mail: the Draft body could be saved as an empty one.
- Bug Fix: E-mail: the autocomplete list could appear in wrong places.
- Bug Fix: Video: the Pause button acted as the Stop button.
- Bug Fix: IM: the custom status composer did not support japanese input.
- Bug Fix: IM: AIR: an exception was thrown when dragging a desktop file into an IM window.
- Bug Fix: Photos: some thumbnails were not displayed in previews.
- Bug Fix: Preferences: Mail Rules: an exception was thrown when double-clicking a column header.
- Bug Fix: Preferences: Mail Rules: User rights restrictions are applied now.
- Bug Fix: Tasks: the reply text was not sent to the Task owner when the Task was accepted.
- Bug Fix: Tasks: an exception could be thrown when a Task was accepted.
- Bug Fix: Twitter: the Refresh Time setting did not work.
- Bug Fix: Phone: a user info was not shown after that user had been split and then merged again.
- Bug Fix: Phone: incoming call value did not accord with the system volume setting.

## 4.0 CommuniGate Pro 5.4.0

- Core: window positions are now stored in the file storage, instead of "local shared objects".
- General: AIR. the Login screen is displayed when the 'Change host' popup is closed.
- Styles: new Paper style has been added.
- Shell: AIR: modules menu scrolling has been improved.
- Shell: the Feedback button has been added.
- General: autoscroll of Prontologs can be disabled now.
- Preferences: the RPOP (e-mail aggregation) management has been implemented.
- Preferences: the Mail Rules management has been implemented.
- Preferences: the Signal Rules management has been implemented.
- Preferences: the Change Password panel has been improved.
- E-mail: the 'Empty Trash' functionality is added
- E-mail: headers with a large number of recipients are scrollable now.
- E-mail: real names in addresses copied from header fields are enclosed within the quotation marks.
- E-mail: AIR: files can be attached by dragging them from desktop folders.
- E-mail: the user Real Name is displayed instead of 'Me' in message headers.
- Voice: the 'All Devices' option for outgoing calls has been added.
- Voice: callDisconnected status messages are displayed now.
- Voice: the Click-to-call function is implemented.
- Calendar: own calendar can be deleted without closing it first.
- Calendar: Weekly views display the current day now.
- Calendar: custom calendar labels are supported now.
- Calendar: multiple addresses can be entered into the attendee field.
- Tasks: the Task module is implemented.
- Files: ACL dialog are improvemed.
- IM: the Preferences button is removed from the IM panel.
- IM: time stamps for offline messages are suppoered now.
- IM: VCards are cached now to reduce network traffice.
- IM: dragged contact display has been improved.
- Music: the Music module has been implemented.
- Photo: the Photo module is implemented.
- Twitter: the functionality has been improved.
- Bug Fix: Core: the Auto Sign-up function did not login after signing up.
- Bug Fix: General: window shadow could block other windows.
- Bug Fix: General: the LTR interface could not be set.
- Bug Fix: General: Pronto could not be loaded in the same browser tab after the Pronto server version has

changed.
- Bug Fix: General: the Clear cache request was not handled correctly.
- Bug Fix: General: the 'Tab' key did not jump to the Password field on the Login page.
- Bug Fix: General: the Auto-login function did not work.
- Bug Fix: General: Pronto could add an incorrect domain name when logging in.
- Bug Fix: General: an arbitrary module could be auto-started after logging in.
- Bug Fix: General: AIR: the Logout item did not work from the tray menu.
- Bug Fix: General: AIR: Logout did not work if message composing was never completed.
- Bug Fix: General: AIR: Notifications did not appear in the non-popup mode.
- Bug Fix: Shell: AIR: notifications created right after logging in were not displayed.
- Bug Fix: Preferences: AIR: the Logout button disappeared after changing the language.
- Bug Fix: Preferences: an exception could be thrown when the Preferences popup was resized.
- Bug Fix: Preferences: the GIPS plugin installation dialog appeared on updating.
- Bug Fix: E-mail: an exception was thrown when deleting an attachment.
- Bug Fix: E-mail: auto-complete did not work for addresses in 'My Domain' area.
- Bug Fix: E-mail: deleting a large number of messages was not possible.
- Bug Fix: E-mail: error #2038 was received when adding attachment using encrypted connection (https://).
- Bug Fix: E-mail: when move a message to the folder, the number of messages did not increase.
- Bug Fix: E-mail: if a filter was applied to INBOX, unread count was not shown inside the EMail module icon.
- Bug Fix: E-mail: message grid scrolling did not work in some cases.
- Bug Fix: E-mail: round brackets in the RTL interface were displayed incorrectly.
- Bug Fix: E-mail: the ctrl-A command did not select all messages after some messages have been deleted.
- Bug Fix: E-mail: auto-complete list was shown beyond the popup borders.
- Bug Fix: E-mail: the 1st message in grid could not be selected after sorting.
- Bug Fix: E-mail: the default font was not applied to new messages.
- Bug Fix: E-mail: unable to forward inline attached message after saving it as a draft.
- Bug Fix: E-mail: font size increase did not have effect on already received messages.
- Bug Fix: E-mail: Preferences: Redirect All mail to' did not work.
- Bug Fix: E-mail: auto-complete list was disappearing after typing several letters.
- Bug Fix: E-mail: more than one address could not be copied from the To: field.
- Bug Fix: E-mail: E-mail subjects were truncated in notifications to "...".
- Bug Fix: E-mail: error was thrown when unsubscribe from a folder with sub-folders.
- Bug Fix: E-mail: messages displayed in a popup window and opened from a filtered view were invalid after removing the filter.
- Bug Fix: E-mail: suscribed sub-folders did not appear in the mailbox tree.
- Bug Fix: E-mail: message headers did not display avatars when users were sending a message to themselves.
- Bug Fix: E-mail: the newly arrived messages were not shown after the Search filter was reset.
- Bug Fix: E-mail: the cursor was jumping to the new line when pasting the text to a plain-text message.
- Bug Fix: E-mail: an exception was thrown when an out-of-stage message was removed.
- Bug Fix: E-mail: messages remained striken out after at attempt to move them failed.
- Bug Fix: E-mail: flagged messages could not be deleted if the 'Show Flagged' filter was applied.
- Bug Fix: E-mail: an undesired visual effect took place when appling the 'Voice Mail' filter.
- Bug Fix: E-mail: .pdf attachments were not shown in some messages.
- Bug Fix: E-mail: certain messages could crash the Flash player.
- Bug Fix: E-mail: a new search did not start when a new filter was being typed.
- Bug Fix: E-mail: the calendar subscription was shown in E-Mail module.
- Bug Fix: E-mail: the Search button did not work in the 'Folder Subscription' dialog.
- Bug Fix: E-mail: some messages remained striked out after moving a large number of messages.
- Bug Fix: E-mail: messages were not removed when the 'Show Flagged' filter was applied.
- Bug Fix: E-mail: the Remove Folder dialog was displayed when renaming a folder.
- Bug Fix: E-mail: the Add Contact function did not work.
- Bug Fix: E-mail: auto-complete did not work if any address has been deleted.
- Bug Fix: E-mail: exception was thown when opening an attached message.
- Bug Fix: E-mail: exception was thrown when dragging messages selected using Ctrl-A.

- Bug Fix: E-mail: Japanese subjects were truncated.
- Bug Fix: E-mail: addresses copy from and pasted into message headers remained formated as links.
- Bug Fix: E-mail: the 'Show HTML' function did not work for html messages.
- Bug Fix: E-mail: exception could be thrown when scrolling messages with embedded graphics.
- Bug Fix: E-mail: the 'Send' button was still disabled after dragging a contact from IM to the "To:" field.
- Bug Fix: E-mail: a message body could be not displayed when reopening the E-Mail module.
- Bug Fix: E-mail: AIR: an image was not shown after dropping a graphics attachment into the "rich-text" editor.
- Bug Fix: E-mail: the Trash folder item counter was not updated automatically.
- Bug Fix: E-mail: pasting a new text into the To: field did not work.
- Bug Fix: E-mail: AIR. some links could not be opened.
- Bug Fix: E-mail: a Subject was missing in a notification if the Subject column was not displayed.
- Bug Fix: E-mail: the '<' symbol within double quotes in To: field was processed incorrectly.
- Bug Fix: Calendar: E-mail folders were not shown in folder subscriptions.
- Bug Fix: Calendar: recurrency exception increased its length after editing the rest of the series.
- Bug Fix: Calendar: the date of an event and its recurrency exception could be in different formats.
- Bug Fix: Calendar: the unsubscribe function did not work.
- Bug Fix: Calendar: the event color changed after the event was clicked.
- Bug Fix: Calendar: the current time marker disappeared when views were switched.
- Bug Fix: Calendar: empty warning notifications appeared when accepting an event.
- Bug Fix: Calendar: the grid and the new event popup used different time formats.
- Bug Fix: Calendar: the date range of an all-day event was not displayed.
- Bug Fix: Calendar: the new calendar default color was black.
- Bug Fix: Calendar: the working hours were not shown int the Today view.
- Bug Fix: Calendar: the divider lines were not shown in Weekly view.
- Bug Fix: Calendar: calendars created with Pronto 3 could be displayed twice.
- Bug Fix: Calendar: An event with a long subject could not be opened.
- Bug Fix: Calendar: saving of an event could fail.
- Bug Fix: Calendar: alarm time was not saved with an event.
- Bug Fix: Calendar: event creation windows could not be resized.
- Bug Fix: Calendar: the Create Folder dialog has been renamed into Create Calendar.
- Bug Fix: Calendar: the Show Attendees function did not work at the first click.
- Bug Fix: Calendar: AIR: the recurrence dialog could be displayed incorrectly.
- Bug Fix: Calendar: events could not be created in some calendars.
- Bug Fix: Calendar: the day of week display was missing from some Weekly views.
- Bug Fix: Calendar: subfolders were not displayed.
- Bug Fix: Calendar: the Month in the navigation bar did not change according to the month selected in the grid.
- Bug Fix: Calendar: alarms could not be canceled.
- Bug Fix: Calendar: an event occurance could not be deleted from recurring series.
- Bug Fix: Calendar: a recurring event could disappear from the grid.
- Bug Fix: Calendar: MacOS: the Event popup was clipped
- Bug Fix: Calendar: the context menu could contain inactive items.
- Bug Fix: Calendar: the Calendar color was not saved.
- Bug Fix: Calendar: the Accept/Tentative/Decline dialog was shown if an attached message contained an Event invitation
- Bug Fix: Calendar: AIR: Pronto could hang after several logouts with Calendar module opened.
- Bug Fix: Calendar: attachments were not saved when accepting an event.
- Bug Fix: Contacts: an exception was thrown on group open.
- Bug Fix: Contacts: some context menus were displayed incorrectly.
- Bug Fix: Contacts: the Whiteboard icon appeared in empty contact fields.
- Bug Fix: Contacts: 'Folder has not been opened' error took place when importing vCard data.
- Bug Fix: Contacts: the contact preview pane could disappear completely after being hidden.
- Bug Fix: Contacts: AIR: the datagrid did not keep selection while dragging.
- Bug Fix: Contacts: autocomplete did not find contact groups
- Bug Fix: Files: Attachment saving to root folder created an extra folder

- Bug Fix: Files: Wrong info dispalyed in uploading popup (if total files sizes >2Gb)
- Bug Fix: Files: When sort by Modified header, arrow sorting sign changed in Size header.
- Bug Fix: Voice: the 'Mute' button was available only for the first conference participant.
- Bug Fix: Voice: if users exited a conference and then joined it again, their infor was displayed incorrectly.
- Bug Fix: Voice: the Volume control did not work.
- Bug Fix: Voice: in some cases invited users did not see all conference participants.
- Bug Fix: Voice: the GIPS plugin install was broken in Safari.
- Bug Fix: Voice: the Dialer was throwing an exception on initialization.
- Bug Fix: Voice: the Call History popup did not close when the call ended.
- Bug Fix: Voice: the 'Plugin not available' message was always shown for AIR installations without a plugin.
- Bug Fix: Voice: AIR: keypad could not be closed.
- Bug Fix: Voice: some incoming calls did not have audio connectivity.
- Bug Fix: Whiteboard: an error occured when closing a whiteboard.
- Bug Fix: IM: User own VCards were added to Contacts multiple times
- Bug Fix: IM: Error occured when delete some non-existent contacts from roster.
- Bug Fix: IM: VCard storing failled when Contacts->IM sync was disabled
- Bug Fix: IM: avatars were not displayed in the Roster if the 'Show contacts in IM' open was switched off.
- Bug Fix: IM: the Calendar Alarm buddy was saved in the Roster when an alarm was received.
- Bug Fix: IM: manual Roster group sorting was broken.
- Bug Fix: IM: input lines in the IM composer were breaking at the cursor position.
- Bug Fix: IM: group chat messages could be displayed in the wrong order.
- Bug Fix: IM: files sent from other domains could not be downloaded.
- Bug Fix: IM: files with names containing the '%' symbol could not be opened.
- Bug Fix: IM: an invalid address could be added to the Roster.
- Bug Fix: IM: japanese text was not sent.
- Bug Fix: IM: the IM sender was automatically added to the recipient's Roster.
- Bug Fix: IM: AIR: dragged contacts were shown away from cursor.
- Bug Fix: IM: AIR: and error could occure when clicking the Pronto icon in tray.
- Bug Fix: Social: error occured when autorefreshing.
- Bug Fix: Twitter: wall scrolling did not work
- Bug Fix: Twitter: followers were not displayed.
- Bug Fix: Twitter: links were not clickable.

**4.0c2** [CommuniGate Pro 5.4c2](#)

- Core: Framework reimplemented. Modular structure: application can be composed of any subset of available modules. The user can define the module set in preferences.
- User interface design reimplemented.
- Styles: Look and feel can be changed on the fly, no relogin needed.
- Styles: new styling scheme, including gradients and bitmaps. A new style editor is provided, no compilation is needed to create a custom style.
- Styles: Each look and feel can now have its own set of icons.
- E-mail: module reimplemented.
- E-mail: simple HTML messages are now displayed directly in web version.
- E-mail: Notifications on new e-mails added.
- Calendar: module reimplemented.
- Calendar" Recurrence exception handling added.
- Calendar: event can be scheduled as a teleconference with the server calling all the participants at the scheduled event time.
- IM: module reimplemented.
- IM: various preferences added regulating display of multiple IM windows.
- IM: several preferences added for buddy list display.
- IM: smileys implemented.
- IM: preferences for conversation formatting added.

- IM: notifications on new IM messages added.
- IM: File transfer implemented.
- Contacts: module reimplemented.
- Contacts: Datasets support added.
- Contacts: custom fields in vCard supported in UI.
- Contacts: Columns management added. "Organization" column is included.
- Telephony: module reimplemented.
- Telephony: each conference participant can now see all the participants.
- Telefony: missing calls notifications added.
- SMS: SMS conversation implemented.
- CRM: Safesforce integration added.
- Whiteboard: module implemented.

# CommuniGate Pro

| **Pronto!** | **Module API** | **History** |

# Pronto! Revision History

- **6.0**
- **4.x**
- **3.15**
- **3.14**
- **3.13**
- **3.11**
- **3.10**
- **3.9**
- **3.8**
- **3.7**
- **3.6**
- **3.5**
- **3.4**
- **3.3**
- **3.2**
- **3.1**
- **3.0**
- **2.x**

**3.15** CommuniGate Pro 5.3.15

- Bug Fix: E-mail: some E-mails could not be forwarded.
- Bug Fix: Voice: the GIPS media plugin could not be installed in Safari.
- Bug Fix: Voice: Call history was not displayed in some cases.
- Bug Fix: Files: Uploading of an empty file caused an error.
- Bug Fix: E-mail: Messages without a subject caused a printing error.
- Bug Fix: Preferences: Password updating could work incorrectly.
- Bug Fix: IM: Links with non-ascii characters did not work.

**3.14** CommuniGate Pro 5.3.14

- Calendar: shared calendars Account name is displayed now.
- IM: offline messages are supported now.
- Mail: dialog formatting colors are more readable now when printed.
- Bug Fix: Calendar: shared events could not be opened after they were accepted.
- Bug Fix: Calendar: an incorrect date was displayed in all-day event invitations.
- Bug Fix: Mail: the "Email File" operation did not work for forwarded emails.
- Bug Fix: Mail: a message body was not properly composed when replying on forwarded messages.
- Bug Fix: Mail: vacation rule end date was processed incorrectly.
- Bug Fix: Mail: Forwarding messages with attachments could fail.
- Bug Fix: Mail: Mail Delivery Notifications were sent when viewing messages in the Sent Items folder.

**3.13** CommuniGate Pro 5.3.13

- General: New buttons with customizable links are added to login screen: "Help & contact" and "Terms & Conditions".
- General: A customizable link is added to the upper left logo.
- E-mail: The Print function now employs the WebUser Interface mechanism.
- E-mail: Deleted mailboxes are moved in Trash now, if the Delete method is set to "Move to trash".
- Bug Fix: the "E-mail me my password" button stayed disabled after incorrect account name was entered.
- Bug Fix: E-mail: Long message lines without spaces were cut when printing.
- Bug Fix: Calendar: All-Day events with zero duration were not displayed.
- Bug Fix: Calendar: some Events in monthly view were not displayed.
- Bug Fix: Core: Pop-up windows were displayed incorrectly on screens with certain resolutions.

**3.11** CommuniGate Pro 5.3.11

- The main navigation panel can contain link-style icons now.
- The logo (on the login page and inside the main framework) works as a customizable link now.
- Styles: More style selections introduced (background image).
- Video Call: the bandwidth control is implemented.
- Voice: the "peerName" attribute of an incoming call is used if the caller's vCard is not found.
- E-mail: the default RPOP server list is added to strings.data.
- E-mail: when the Japanese language is selected, the default font in e-mail Composer is set to Gothic.
- Bug Fix: Layout: the Banner machanism used incorrect timeout values.
- Bug Fix: Login: a user could not login if a non-existent Skin was selected.
- Bug fix: IM: the module did not start if no E-mail has been opened yet.
- Bug Fix: E-mail: links right after a line break were not recognized.
- Bug Fix: E-mail: reply messages got additional default receipients if filtering was used.
- Bug Fix: E-mail: scrolling the auto-complete drop-down list could close it.
- Bug Fix: Mail/Calendar: Task attachments could cause exceptions.
- Bug Fix: Voice: Media plugin did not install automatically in Microsoft Internet Explorer.
- Bug Fix: Dialer: some ringback tones could be not played.
- Bug Fix: Dialer: the "find contact" operations could loop.
- Bug Fix: Dialer: the peer's address was not always displayed.
- Bug Fix: Calendar: some invitations were not shown.
- Bug Fix: Calendar: view switching was broken when the animations were switched off.
- Bug Fix: Calendar: month-long events for a previous month were duplicated.
- Bug Fix: Calendar: events were duplicated on second import.
- Bug Fix: Social: the Facebook authentication did not work in Microsoft Internet Explorer.

**3.10** CommuniGate Pro 5.3.10

- Voice: AIR: the CGP plugin support is added.
- Voice: the "Alternative Contact" and "Home PSTN Number" items are added to the "Call Control" panel.
- Rules: multiline parameter fields are supported now.
- E-mail: the "Empty Trash" button can be removed using the strings dataset.
- E-mail: the default font for Japanese messages is changed from Mincho to Gothic.
- E-mail: a message without a To address, but with Cc/Bcc address(es) can be sent now.
- Twitter: the Sign In button added.
- Twitter: the new autentication method is supported now.
- Social: the Facebook OAuth2 authentication is supported now.
- Bug Fix: Contacts: the first contact could not selected in some cases.
- Bug Fix: the Twitter module could cause an error at re-login.
- Bug Fix: E-mail: inline images were not displayed in drafts.
- Bug Fix: E-mail: resending an attachment could cause an error.

- Bug Fix: E-mail: AIR. padding for html and plain-text messages was not the same.
- Bug Fix: E-mail: an HTML part was attached to an empty plain-text message.
- Bug Fix: IM: an exception could occur when the user is kicked out of an IM chatroom.
- Bug Fix: Voice: media touch tone did not work under Mac OS.
- Bug Fix: Voice: plugin did not install properly in some cases.
- Bug Fix: Twitter: twit deleting could cause an exception.
- Bug Fix: Twitter: twits with special characters could not be sent.
- Bug Fix: Twitter: twits marked as 'favorite' could not be deleted.

## 3.9 CommuniGate Pro 5.3.9

- AIR: Phone functionality added. Native installers for Windows and Mac platforms implemented.
- Pronto AIR now installs into the Applications root folder (MacOSX, MS Windows).
- Media Plugin: the installation procedures have been updated.
- The auto-login "random" URL parameter is supported now.
- Rules: the Vacation "end date" field is supported now.
- UI: Some screen items have been renamed.
- Social: the Facebook interface has been migrated to new data model.
- Social: sending an image without text is implemented.
- Bug Fix: Calendar: Events could not be created by double-clicking on any current week day.
- Bug Fix: Calendar: Hint time was always displayed in the 24hour format.
- Bug Fix: Contacts: Importing a vCard with an empty email address was throwing an exception.
- Bug Fix: Contacts: Preset avatars were added to vCards without their image type.
- Bug Fix: Contacts: Dragging or double clicking the first contact in a filtered grid was throwing an exception.
- Bug Fix: Contacts: A comma was displayed after a job title in the summary pane.
- Bug Fix: E-mail: Inline images were not displayed in forwarded messages when forwarded as attachment.
- Bug Fix: E-mail: Inline images were not displayed under MacOS X.
- Bug Fix: E-mail: The first address in a filtered grid could not be selected when redirecting an email.
- Bug Fix: E-mail: AIR. Animations did not work in the Composer window.
- Bug Fix: Files: ACLs created via the WebUser interface were not displayed.
- Bug Fix: Core: the "bye" command was not always sent at logout.
- Bug Fix: Core: the Japanese layout was reset to the English one after an OS window was opened.
- Bug Fix: Core: the context menus were not localized after relogin.
- Bug Fix: IM: Adding a participant to a chatroom could throw an exception.
- Bug Fix: IM: Some context menu items were not displayed.
- Bug Fix: Music: Double-clicking a music file did not start playing it.
- Bug Fix: Twitter: Adding a new account dialog had a wrong button label.
- Bug Fix: Twitter: The ampersand symbol was breaking a twit.
- Bug Fix: Twitter: A word with a '#' tag was not displayed when sending tweets.
- Bug Fix: Twitter: Tagged words could become non-clickable.
- Bug Fix: Voice: Calls could overload the CPU and result in crashes.
- Bug Fix: Voice: the Call item was disappearing from the IM menu after saving Preferences.
- Bug Fix: Voice: Logging out while a call line was active was throwing an exception.
- Bug Fix: Telephony: The toolbar buttons in the Telephony and History views could be left enabled.
- Bug Fix: Voice: Clicking the 'Join to' in an incoming call window could throw an exception.
- Bug Fix: The AIR2 Log window could cause crashes.
- Bug Fix: Voice: the Dialer module could leak memory.
- Bug Fix: Voice Mail: the Voice Mail player did not work over slow connections.
- Bug Fix: Voice Mail: the context menu for .wav attachments was not working.
- Bug Fix: IM: Mouse wheel scrolling worked incorrectly in some Flash Player versions.
- Bug Fix: IM: AIR: scrolling worked incorrectly.
- Bug Fix: IM: AIR: the Clipboard context menu was not displayed.
- Bug Fix: Non-email folders were displayed in RPOP preferences.

## 3.8 CommuniGate Pro 5.3.8

- AIR: the login page 'Save password' option can be hidden using a strings.data string.
- E-mail: the X-Color header for spam messages is supported.
- E-mail: the Mail Rule editor is improved.
- Preferences: the Vacation Message 'Ends date' option is supported now.
- Bug Fix: AIR: the current IM window lost focus when a new window was open for an incoming IM.
- Bug Fix: AIR: the 'mailto' address link did not open a new compose window.
- Bug Fix: AIR: E-mail: message window scrollbar was displayed incorrectly.
- Bug Fix: AIR: the Compose window could be closed without a confirmation dialog.
- Bug Fix: Voice: the 'Call' window size was changed incorrectly when new contacts were added.
- Bug Fix: Voice: a 'bump' sound could be heard at logout.
- Bug Fix: Voice: the Dialer could continue ringing when the call had been answered.
- Bug Fix: Styles: the time slice bar was invisible within the Dark Flame style.
- Bug Fix: Videochat: the VideoCall window was not displayed if no IM window had been already opened.
- Bug Fix: Calendar: the Calendar module could throw exceptions within the 'Classic' style.
- Bug Fix: Contacts: the remotely retrieved Contacts (vCards) could be incorrectly merged with the existing ones.
- Bug Fix: E-mail: uploading of a large attachment could not be canceled.

## 3.7 CommuniGate Pro 5.3.7

- Phone: if the Outgoing Calls preference is set "All devices", Pronto! click-to-call is enabled even if the media plugin is not loaded.
- IM: AIR: IM window for a new IM chat appears in open, not minimized state.
- Bug Fix: Calendar: Adding a folder to the "Subscriptions" folder caused an exception.
- Bug Fix: Calendar: Alerts did not work after relogin.
- Bug Fix: E-mail: the message header To: field could be displayed incorrectly.
- Bug Fix: E-mail: Some hints were not removed on Compose window closing.
- Bug Fix: Rules: Some Junk Mail rules were set incorrectly.
- Bug Fix: IM: Some strings were not localized.
- Bug Fix: Phone: extra "CallUpdate" was issued when created a client conference.

## 3.6 CommuniGate Pro 5.3.6

- Kernel: the Flash Policy File is now loaded explicitly eliminating 3 sec timeout.
- Bug Fix: E-mail: a draft message saved in the HTML formal could not be re-saved as Plain Text.
- Bug Fix: E-mail: HTML styles were removed when switching formats from HTML to Plain text even if the operation was canceled by the user.
- Bug Fix: E-mail: Internal error notification was given sometimes when no error occurred.
- Bug Fix: E-mail: Renaming of INBOX caused an exception.
- Bug Fix: E-mail: Display of e-mail had unnecessary blinking of the e-mail body field.
- Bug Fix: 'Notes' and 'Tasks' items were displayed in 'New folder' dialog popup.
- Bug Fix: Some strings were not localized.
- Bug Fix: RSS: Some fields were not displayed.
- Bug Fix: E-mail: Renamed 'Inbox' folder was not displayed in RPOP folder list.
- Bug Fix: Social: Some posts were not displayed correctly.
- Bug Fix: Social: Facebook sessions were not persistent.
- Bug Fix: Voice: Recorder did not work in some cases.
- Bug Fix: Voice: Keypad did not work when 'media' sending method was used.
- Bug Fix: E-mail: Exception was thrown at re-login if a voicemail message was the first one.

## 3.5 CommuniGate Pro 5.3.5

- General: the "Back" browser button is handled correctly now.

- Calendar: iCal-generated invitations are supported now.
- Calendar: preference settings for Server-Side Alert processing have been added.
- Video Call: Signal protocol was reimplemented using special IM messages.
- E-mail: Link parsing has been improved.
- Twitter: Retweets are now shown on the Wall
- Bug Fix: Calendar: Events in a Subscribed calendar were not displayed if the calendar name was the same as the Main Calendar name.
- Bug Fix: Calendar: Simple events were saved in UTC timezone (daylight saving time problems).
- Bug Fix: Calendar: zero-duration events in Subscribed calendars were not displayed.
- Bug Fix: Contacts: the MessagePart URL to the raw contact data was built incorrectly if the Contacts folders name contained non-Latin symbols.
- Bug Fix: E-mail: the priority tag could not be removed from draft messages.
- Bug Fix: AIR: the pop-up window maximize button could crash the application if a large monitor was used.
- Bug Fix: IM: avatars were not displayed in the conversation field in some cases.
- Bug Fix: Preferences: the 'Simultaneous Ring' option in the Call Control panel could not be switched off.
- Bug Fix: Voice: there was an extra CallUpdate XIMSS request when an outgoing call was established.
- Bug Fix: Voice: the 'Add me' conference function did not work if outgoing calls were configured to go via "All Devices".
- Bug Fix: Twitter: clicking a @username link displayed all the twits posted by that user instead of reply-twits to that user.
- Bug Fix: Twitter: search did not always display relevant results.
- Bug Fix: Twitter: logging out of Pronto while waiting for authentication to complete could cause an exception.

## 3.4 CommuniGate Pro 5.3.4

- Mail: Voicemails are played directly in Pronto! (no external media player needed).
- Mail: an audio feedback is provided if a user command is not accepted.
- Calendar: Normal and military time is now shown according to the language preferences.
- Calendar: Current day and time visual indications added.
- Calendar: Minuscule events are displayed in a more appropriate way.
- Calendar: Events with zero duration are displayed now.
- Calendar: the Event editor "Send requests" check box is disabled by default if the event is not in the default calendar.
- Contacts: selected Directory contacts can be opened with a double click.
- IM: the context menu now contains the "Call xxxxxx" items for all roster item E-mails and phone numbers.
- Modules: the "HiddenProntoModules" Account setting is supported now.
- Phone: the Test sound button has been added to the Create Call window.
- Files: the folder tree in the Files, Photos, Music modules displays file directories only.
- Kernel: Spanish and Polish language files have been added.
- Video chat: the Enable/Disable Video button has been added.
- Twitter: the Login/Password form has been added to the navigation panel.
- Bug Fix: Calendar: the Where and When event labels in the tooltip were not localized.
- Bug Fix: Calendar: any 'Monthly by Weekday' recurrence was reset when an event was updated.
- Bug Fix: Calendar: time intervals in the Time Slice bar and in the Main Calendar area could differ.
- Bug Fix: Contacts: clicking 'Account Name' in the 'Shared Contacts' popup could cause an exception.
- Bug Fix: Mail: E-mails with certain MIME structures were not displayed.
- Bug Fix: Mail: total message count was not increased on new E-mail arrival when some other module was active.
- Bug Fix: Mail: the Print button did not work for E-mails without a subject.
- Bug Fix: Mail: sent message was shown in other open E-mail compose windows.
- Bug Fix: Mail: composed E-mails had no signature when a message was forwarded as an attachment.
- Bug Fix: Mail: pressing the down-arrow key after inserting of a large image could cause an exception.
- Bug Fix: Files: extra ACL notifications were sent if a shared folder had subfolders.
- Bug Fix: Files: deleted files were still shown after refreshing the shared folder view.

- Bug Fix: Files: the Subscription button could stop working.
- Bug Fix: Files: folder unsubscription did not work in some cases.
- Bug Fix: Styles: Some styles in the Preferences window were not readable.
- Bug Fix: AIR: Pronto! could not be started in some cases.
- Bug Fix: Video Chat: Minimizing of the videocall popup could cause an exception.
- Bug Fix: the Password settings were not hidden if the Password Modification and Password Recovery settings were disabled.
- Bug Fix: Social: the Facebook "offline access" permissions could fail.
- Bug Fix: Video: Video was disabled after an attempt to use an unsupported format.
- Bug Fix: Phone: the Media plugin download link for the Chrome browser was incorrect.

**3.3** CommuniGate Pro 5.3.3

- Login: logoutURL parameter is documented.
- Kernel: internal logs are stored in the plain text format now.
- Kernel: now the internal log manager can perform search operations.
- Kernel: now the internal log manager can perform the Send operation.
- Phone: Chrome browser is supported now.
- Phone: Firefox is now using the *.xpi Plugins in every OS.
- Help: the old help files are removed from "Pronto- server skin.
- Help: the Help URL can be specified individually for each language using the HelpBaseURL string in strings.data
- Bug Fix: the Call and Feedback buttons were not displayed in certain cases.
- Bug Fix: Mail, AIR: the HTML Renderer could set the scroll bar position incorrectly.
- Bug Fix: Mail: an error occurred when applying English check spelling to non-ascii messages.
- Bug Fix: Mail: an spellchecker error occurred if the message last word was misspelled.
- Bug Fix: Mail: the Busy clock could stay on screen if a new message arrived to an empty folder.
- Bug Fix: Mail: the message selector could jump back and force if server connectivity was slow.
- Bug Fix: Twitter: new user data was displayed incorrectly.
- Bug Fix: Contacts: the first contact in the filtered address book could not be selected.
- Bug Fix: Contacts: clicking on the Account Name label in the Shared Contacts popup resulted in an exception.
- Bug Fix: IM: tooltip presence indicator did not change when the buddy presence changed.
- Bug Fix: Log: the Send Log confirmation dialog could be shown behind other windows.
- Bug Fix: Files: Subscription button did not work in some cases.
- Bug Fix: Files: Deleted files were still shown after refreshing the shared folder.
- Bug Fix: Files: Subfolder could inherit its parent folder rights incorrectly.

**3.2** CommuniGate Pro 5.3.2

- Kernel: the Adobe Text Layout Framework library is updated.
- Settings: the Animation preference is added.
- Settings: now the dialer preferences "tooltip" displays the full Media Plugin version info.
- Dialer: now a special signal indicates an incoming call when the Dialer is active (instead of the standard ring sound).
- Bug Fix: Kernel: window animation debug info added.
- Bug Fix: Mail: filter field was editable while Flagged checkbox was selected.
- Bug Fix: Mail: E-mail message details (To, Cc, Bcc fields) were cleaned up when the message was updated in a "search" view.
- Bug Fix: Mail: the Junk setting processing was incorrect.
- Bug Fix: Dialer: an error occurred if a media leg was created in CommuniGate Pro Media Plugin.
- Bug Fix: Dialer: the "mute" button did not work for the organizer of a client-side conference.
- Bug Fix: Dialer: dragging user avatars into a conference could fail.
- Bug Fix: Dialer: unmute did not work after hold-resume operation.
- Bug Fix: Dialer: an extra window was added when an incoming call joined an existing line.

- Bug Fix: Contacts: some sorting operations could cause exceptions.
- Bug Fix: AIR: clicking the pop-up window "maximize" button could cause a crash.
- Bug Fix: AIR, Video: Video-file playing looped.
- Bug Fix: Roster: adding new Group member threw an exception.
- Bug Fix: Help: some links were broken.

**3.1** CommuniGate Pro 5.3.1

- Contacts: Yomi fields for family name, given name, company and unit name added (when the Japanese language is selected).
- Contacts: Groups: email icon is added to each contact in group preview.
- Contacts: Groups: email is now composed using all contacts selected in list.
- Mail: Mailbox ACL notification is sent using the text/plain format.
- Mail: mail preview scrollbar scrolling is a bit smoother now.
- Mail: visual feedback for a busy folder (search/sort) is added.
- Mail/Calendar: if a MONTHLY recurrence does not specify recurrence day, then the day of event creation is assumed.
- IM: chatroom members avatars are loaded now (via the chatroom).
- IM: the "message is composing" sending period is increased to 10 seconds.
- IM: the chatroom window input field is disabled if the voice right is revoked.
- IM: users are notified when they is disconnected from a chatroom.
- Login: the random cypher string from session response is now used with secure sockets and HTTP Authentication.
- Bug Fix: Mail: sometime the busy cursor was left on screen when all operations were completed.
- Bug Fix: Mail: Reply, Reply TO All did not work when the preview pane was closed.
- Bug Fix: Mail: Empty search results didn't change number of messages in the header.
- Bug Fix: Mail: when new messages were arriving, the current mailbox grid selection was updated incorrectly.
- Bug Fix: Mail: incorrect URLs were used for inline images.
- Bug Fix: Mail: the mailbox grid was not re-enabled after search yielding no results.
- Bug Fix: Mail/Contacts: the autocomplete function did not show Groups.
- Bug Fix: Mail/Calendar: events without an organizer could cause exceptions.
- Bug Fix: IM: chatroom presence could be shown incorrectly.
- Bug Fix: IM: when contact is renamed its name was not updated immediately.
- Bug Fix: IM: Conference invitations did not beep.
- Bug Fix: IM: users without the "Real Name" settings did not see their own avatars.
- Bug fix: IM: the "message is composing" stanza could be not recognized.
- Big Fix: Contacts: Groups: E-mail addresses were parsed incorrectly when adding a new contact to a group.
- Big Fix: Contacts: Groups: the "caller" tooltip is renamed to "call" in the Group preview.
- Bug Fix: Contacts: the "To" Field auto-complete did not work.
- Bug Fix: Contacts: directory view scrolling could cause exceptions.
- Bug Fix: Contacts: unsortable directory views were shown as sortable.
- Bug fix: Contacts: mailbox re-sorting used incorrect field names.
- Bug fix: Contacts: auto-complete function inserted the contact type instead of the contact E-mail address.
- Bug fix: Contacts: the Compose button did not work in the Contact window.
- Bug Fix: Contacts: if contact popup is opened before contact body is loaded, vCard is not updated.
- Bug Fix: Mailboxes: mailbox synchronization was not performed if two modification notifications were received.
- Bug Fix: API: the monkeylib.swf was not included.
  - �

**3.0** CommuniGate Pro 5.3.0

- Bug Fix: Preferences: default values for special folders weren't used in case of empty preference.
- Bug Fix: Mail/Contacts: mailbox icons and labels weren't displayed in the tree if draftsBox wasn't set.

**3.0c3** [CommuniGate Pro 5.3c3](#)

- IM: Chat invite accepting preference implemented
- IM: "Add person" button implemented in chat window
- Files: Shared file path in ACL notification is not shown as link any more
- Profile: Added a warning on save if user did not fill out the File As field in his profile
- Date and time display in header: dayAndDateAndTime template is now used
- E-mail: Automatic Preview option implemented
- Voice: Pronto! now detects whether media plugin installed of not, and if not provides the download link
- Voice: Mute/unmute button added to the phone window
- Preferences: Signature window can be expanded on double click to allow editing long signatures
- Social module UI improved
- Social module: It is now possible to add images to post to Flickr
- Login: autologin option is shown as link when user is allowed to enter login/password, otherwise it's shown as button.
- Bug Fix: IM. Avatars were not loaded after relogin
- Bug Fix: IM. User could not scroll the IM chat to the top messages if another participant was typing message at the same moment.
- Bug Fix: Messages were not marked read on opening when preview pane was disabled
- Bug Fix: avatars were not shown in Phone window in some case
- Bug Fix: wrong cursor behavior in e-mail compose window
- Bug Fix: Changes made in address fields were not saved in draft e-mails
- Bug Fix: Multi chat history entries had strange fields for participants and subject
- Bug Fix: Save as field was not shown in contacts in some cases
- Bug Fix: Ctlr+C hotkey did not work for copying for the 1st time
- Bug Fix: Wrong sharing link was generated for the file uploaded right into the root private folder
- Bug Fix: it was impossible to uncheck all the access rights in the ACL in Calendar module
- Bug Fix: Wrong message status icons were shown in some cases
- Bug Fix: Preferences->Call Control ->Simultaneous ring->Number preference was not saved.
- Bug Fix: Calls history was not shown after sorting
- Bug Fix: Wrong sharing link was generated to a newly uploaded file
- Bug Fix: drafts could be deleted before confirmation on message sending was received
- Bug Fix: Unsubscribe form email folder caused unsubscription from folder in Calendar.
- Bug Fix: Situation of logout Pronto while refresh is not completed is now handled
- Bug Fix: Message autosaving was taking focus from currently opened windows

**3.0c1** [CommuniGate Pro 5.3c1](#)

- On-line help is implemented.
- New main navigation is implemented, navigation location/style is a user preference now.
- Module development API and examples are available now.
- New Twitter support module is implemented.
- New Social networks crossposting/search/view module is implemented.
- Styles: 2 new styles added, old basic style is renamed into "classic".
- User Profile (vCard) storage (including the avatar) is supported now.
- Email: External accounts aggregation.
- Email: new rich text editor supports inline images insert.
- Email: right-to- left languages support implemented.
- Email: the Text Editor HTML export is browser compatible now.
- Email: spell checker works in HTML mode now.
- Email: dialog formatting (reply quotation) is implemented.
- Email: preview panel can be disabled: drag the divider between the mail grid and the preview to the bottom.
- Email: new preference "Mark read after" - in sec.The value "Disabled" means the messages are marked manually. When E-mail: "Unread" filter in the mail grid added. Only visible when "Mark as read" option value

is "Disabled".

- E-mail: "Flagged" filter in mail grid is implemented.
- Calendar: new implementation with improved look and feel.
- Calendar: client-side alerts are implemented.
- Calendar: work week customization/week start option are implemented.
- Calendar: the work-hours background highlighting is implemented.
- Calendar: private events are supported now.
- Calendar: unlimited calendar display views are supported now.
- Calendar: access to external (ICS, WebCAL) calendars is implemented now.
- Calendar: calendar publishing is implemented now.
- Contacts: Contact groups are supported now.
- IM: Multi-part chats (groupchat, chatrooms) are supported now. Ad-hoc multipart chats (conferences) are supported.
- IM: Chatroom discovery on local and external XMPP servers is supported now.
- IM: peer vCards are retrieved and merged with the default Contacts folder data.
- IM: the Roster panel is redesigned to include avatars and extended status information.
- IM: Conversation window is redesigned. Peer vCard data can be displayed.
- IM: New functions are added to the pop-up (right click) menu in the Roster panel.
- Files: shared folders and ACL management functions are supported now.
- Video chat: the peer-to-peer communication is implemented. Requires an Adobe FMS-compatible server.
- Music: play lists are displayed in the folder tree now.
- Music: play lists are updated immediately when file sets are modified.
- Music: New sharing options: music player link, music player embed code (for blogs and web sites)
- Banners: images (jpg, gif, png), SWF, and html text can be displayed now.
- Banners: floating banners in selected location are supported now.
- Banners: user actions (dialer open, module open) can display banners now.
- Banners: the Account/Domain names are passed to the banner system to enable per-user customization.
- Photo: list, previews, and slider viewers are implemented.
- Photo: full-screen slide show is implemented.
- Photo: context menu and tool bar operations on files and folders are implemented (copy, move, rename, upload, download, sharing).
- Video: previews are implemented.
- Video: operations on video files and folders are implemented: copy, move, rename, upload/download, sharing.
- Video: video files listing is implemented.
- News module: customizable refresh rates are implemented.
- Voice: Phone module is redesigned.
- Voice: an optional Media Plugin based on GIPS engine is available (requires a separate License Key)
- Voice: Custom ring back is supported.
- Voice: the Dialer interface preference is added.
- Voice: support for client-side conference is implemented.
- Voice: the Input/Output sound device preferences are implemented.

## CommuniGate Pro

| **Pronto!** | **Module API** | **History** |

# Pronto! Revision History

- **6.0**
- **4.x**
- **3.x**
- **2.7**
- **2.6**
- **2.5**
- **2.3.1**
- **2.3**
- **2.2**
- **2.1.3**
- **2.1**
- **2.0**

**2.7** [CommuniGate Pro 5.2.11](#)

- Contacts: custom full name element order is supported now.
- Contacts: the reversed address element order is supported now.
- Contacts: the "avatar image" container is scaled to image size now.
- Music: the player UI has been changed.
- Files: new files can be uploaded by dragging them from the desktop onto the file grid (AIR version only).
- Styles: the "Twilight" style is added.
- Bug Fix: the vertical scrollbar was not shown in some cases when a folder was expanded.
- Bug Fix: File upload failed if the file already existed.
- Bug Fix: for some languages the Delete option was missing in the E-mail message context menu.
- Bug Fix: file download was not working when running under the Flash player version 10.
- Bug Fix: shared folders were not selectable in the "Copy to" dialog box.
- Bug Fix: the News module crashed on incorrect dates.
- Bug Fix: a message was not marked as "answered" if the Mailbox view was a "filtered" one.
- Bug Fix: the Additional Buttons were not recreated on re-login.
- Bug Fix: the free-busy info popup was sized incorrectly.
- Bug Fix: on purging deleted messages, the message grid selected element/area was moved down.
- Bug Fix: the messageSubmit requests contained message body duplicates.

**2.6** [CommuniGate Pro 5.2.10](#)

- Customization: custom logos can be displayed on the login screen and in the main view, rotating with the "Powered by CommuniGate Pro" logo.
- Video: the full screen mode implemented
- Settings: the Mail Rule editor now checks if the user is allowed to update the Rule.
- Dialer: the "dial functions" settings are stored in the standard format now.
- Styles: a preloader container class added to replace the standard Flash preloader on autologin.
- Files: files can now be moved to a different folder using drag-and-drop.
- Bug Fix: Login: LoginOptions. The ForgotPassword Login Option was ignored.
- Bug Fix: Core: the WSSPMacroInterpreter incorrectly processed URIs.
- Bug Fix: Core: Strings were loaded from the default Web Domain Skin instead of the default Pronto Domain Skin.
- Bug Fix: E-Mail: Contact groups were not expanded when composing a message
- Bug Fix: Settings: Some Simplified Signal (Call) Rules were displayed incorrectly.
- Bug Fix: Selecting contacts from Directory Address Books. caused an exception.
- Bug Fix: Contacts: the Create/Rename/Delete/Import operations were incorrectly enabled for Directory Address Books.
- Bug Fix: addresses selected in Address Books pop-up window using double-clicks were always added to the "To" field.
- Bug Fix: File uploading and file sharing did not work with the Flash Player v.10

## 2.5 [CommuniGate Pro 5.2.9](#)

- Loading: preloader is shown until the default skin is loaded/failed to load (both AIR and Flex).
- Kernel: the <connect/> XIMSS "features" element is processed now (it sets the default port and connection type).
- Kernel: the client-side Log is available now.
- Customization: the "Forgot password" link is now controlled with the "LoginOptions" 'string'.
- Customization: Additional Buttons now support URL completion and variables in %%...%%: ID (session id), accountName, domainName, fullAccountName, realName.
- Contacts: Save and Close button are added.
- Contacts: Contact Groups can be viewed and e-mails can be sent to the groups.
- Files: Copy, move, delete operations added for files and folders.
- Files: file types icons added.
- Files: folder sharing now provides 2 types of links: File List and Directory.
- IM: IM preferences moved to a separate panel, Auto IM History Mailbox can be changed now.
- Video: the full-screen mode is implemented.
- AIR: the Print function is now handled using browser, producing results identical to the Web version.
- E-mail: Processing of large messages is improved.
- E-mail: Tabbing works in the text editor now.
- E-mail: Messages with only a text/html body are processed better now.
- E-mail: The Sent Items folder is created when user sends message for the first time and the folder is not found.
- E-mail: The Compose Format (Plain/HTML) setting is implemented.
- Conference: the conference name selector combobox removed.
- Bug Fix: Settings: a Dictionary address book could be selected as the default Contacts folder.
- Bug Fix: Files: the nodule crashed after several open/close tree actions.
- Bug Fix: AIR: If the AIR version was installed for the first time ever, some labels on the login screen were missing.
- Bug Fix: Login: login variables were not URL-escaped when used with HTTP Binding.
- Bug Fix: Email: the Print button was not shown for attached messages.
- Bug Fix: IM History: some history messages had empty subjects.
- Bug Fix: E-mail: the letter "W" was clipped in grid and message preview subject.
- Bug Fix: New Event and New Contact menu items were mixed.
- Bug Fix: the File sharing link used an incorrect URL domain.
- Bug Fix: Core: Memory leak removed in ExtendedPanelLite.

- Bug Fix: E-mail: auto-complete did not work in some cases.
- Bug Fix: E-mail: the Save message operation removed the selection and preview in Drafts.
- Bug Fix: E-mail: a Message body in the preview pane was shown multiple times after repeated search operations.
- Bug Fix: Settings: extra lines were added when saving a vacation message.
- Bug Fix: Spell checker: messages were scrolled to bottom on each correction.
- Bug Fix: Spell checker: the apostrophe symbol was processed incorrectly.
- Bug Fix: Spell checker: manual text editing was breaking spell checker result display.
- Bug Fix: Settings: the ^N macro was not processed at the beginning of a reply header.
- Bug Fix: the DSN/MDN functionality was broken.
- Bug Fix: Contact: avatar uploading was not working.
- Bug Fix: File sharing: links to files were not URL-encoded.
- Bug Fix: Files: the module could be displayed in the central panel.
- Bug Fix: an IM window did not open for Contact items not found in the Roster.
- Bug Fix: VideoMail composer used a wrong URL domain name.
- Bug Fix: Dialer: the window did not stop blinking when call was canceled.
- Bug Fix: Dialer: the window could not be closed in "inactive" states.
- Bug Fix: Settings: the Delete method was not put in effect until the next login.

**2.3.1** [CommuniGate Pro 5.2.8](#)

- FILES: the File manager is implemented.
- Video: the video player layout is improved.
- E-mail: navigation tree and message grid processing is improved.
- Additional button labels are localizable now.
- IM: the History format is improved.
- Bug Fix: Default ProntoSkinName preference was not stored correctly.
- Bug Fix: a context menu for the Files navigation tree was not shown.

**2.3** [CommuniGate Pro 5.2.7](#)

- E-mail: now the available Rule actions are displayed according to the Account rights
- E-mail: the Detach Attachment(s) functionality is implemented
- IM: IM history saving functions are added.
- IM: Ctrl + Enter inserts a line break
- IM: a preference for the IM window Font size is implemented.
- IM: window part sizes can be modified now.
- Customization: login page language menu can be removed now.
- Customization: Custom buttons (linked to some URLs) can be shown on the top toolbar.
- Telephony: Telephony tab icons are changed.
- Dialer: DTMF beep-sound are added.
- AIR: new version check functionality is implemented.
- Calendar: now the Event subject is displayed next to the time label if there is not enough space to display it on a new line.
- Login: now the clear text login method can be used over HTTPS connections (HTTP Binding).
- Bug Fix: Messages with an empty subject could not be forwarded
- Bug Fix: pre-login domain was not correctly set
- Bug Fix: HTTP Binding didn't work in IE7 because of caching problems
- Bug Fix: Pronto crashed sometimes when user entered "dialer" tab in the preferences window.
- Bug Fix: My Stuff: Not all modules were removed from navigation if more than one module was disabled.
- Bug Fix: links in HTML letters in AIR version work now.
- Bug Fix: Calendar: If user had no calendars at all, Pronto crashed when entering Calendar tab.
- Bug Fix: Calendar: AddressBookPopup crashed if contact without email was selected when closing the window
- Bug Fix: AIR: Application is forced to exit when main window is closed.

- Bug Fix: multiple signalBind commands were issued on login
- Bug Fix: Dialer window did not open of first click
- Bug Fix: update dialer playing wav files (now most of them are playing using media plugin)
- Bug Fix: pronto was crashing while switching from music to video.
- Bug Fix: 2 compose windows were open on email context menu click "Forward" or "Reply".
- Bug Fix: Calendar: All-day events were displayed as mere line.
- Bug Fix: Calendar: Contacts and Calendar folders were not created automatically in some cases. Default contacts folder is now created on first login now. Default calendar folder is created on opening Calendar tab for the first time, or by Accepting/Declining appointment message.
- Bug Fix: Top-level pure folders were not displayed in Calendar and Contacts tabs.
- Bug Fix: Files: Attachment or file downloading in Flash Player 10 was broken - a security error was thrown.
- Bug Fix: Pronto! was hanging on login because of selected calendar handling
- Bug Fix: Events were fetched twice on calendar tab opening
- Bug Fix: Events were not fetched and displayed when display range was changed for not merged view, then calendar selection changed
- Bug Fix: In some cases after login the folder title "INBOX" was not shown.

**2.2** CommuniGate Pro 5.2.6

- The new file structure is introduced: modules and styles are implemented as Skins; styles, logo/images, sounds, modules can be modified using the Skin customization methods.
- Style development guide and helper style viewer program is included.
- Telephony: Support for Media plugins under non-IE browsers is implemented.
- VideoMail: now the recorder uses the AMF0 protocol (compatible with Adobe Flash Media Server v2).
- VideoMail: now recorded videomail files are stored in the Account File Storage (optional).
- VideoMail: upload progress bar has been added.
- Calendar: Free/Busy attendee status display has been added.
- Calendar: an attendee status can be changed manually (using the Attendee Status window).
- Calendar: an event busy status can be changed.
- Calendar: the Event Subject is displayed along with the Event timestamp.
- Music: now the Refresh function obtains all music files from Server and checks the playlist meta-file info.
- Photo: Slide show functionality added.
- Mailbox Tree: the drag-and-drop operation with Ctrl-key pressed copies messages.
- Preloader changed to a more generic one.
- News: HTML preview in added to the AIR version.
- Bug Fix: Mailbox Tree: the Drag-and-drop operation did not expand branches.
- Bug Fix: Mailbox Tree: unselectable Mailboxes caused display problems.
- Bug Fix: Contacts: Dragging contact from Directory to folder did not work.
- Bug Fix: Contacts: exception was raised if an E-mail address contained a malformed xml.
- Bug Fix: Contacts: Directory folders were added every time the mailboxRead operation was performed.
- Bug Fix: VideoMail: it was possible to start recording even if the camera failed to initialize.
- Bug Fix: Videomail: FMS connections always failed due to an incorrect timeout.
- Bug Fix: Message body disappears when the html/plain option was changed.
- Bug Fix: Music: the Playlist copy button actually worked as the "move" button.
- Bug Fix: Music: when a new folder was selected, meta data could not be refreshed.
- Bug Fix: Music: the Stop button did not stop loading.
- Bug Fix: Mail: slow connections to the Server could result in incorrect UIDs being sent.
- Bug Fix: Mail: messages with multiple Subjects and/or From fields were processed incorrectly.
- Bug Fix: IM: the Group menu did not show all available Groups.
- Bug Fix: Conference: loading the module could cause exceptions.
- Bug Fix: the Alert window layout was broken.
- Bug Fix: the Folder settings panel did not work.
- Bug Fix: file downloading did not work if the file name included white spaces.
- Bug Fix: when file uploading was interrupted, the file names were not removed from the uploaded file list.

- Bug Fix: multiple files upload could result in error reports.
- Bug Fix: expanding a folder caused the scroll bar to reset to the top.

**2.1.3** CommuniGate Pro 5.2.5

- XIMSS HTTP Binding implemented (support for HTTP proxies).
- Mail: the Basic Video Mail feature is implemented.
- Mail: Attachment management is implemented.
- Mail: now "mailto: " links open a Composer window.
- Mail: attachments are now visible in messages incorrectly composed with the Apple Mail client.
- Contacts: contact data (vCard) import is implemented.
- Calendar: calendar data (.ics and .ical files) import is implemented.
- Files: private folders/files sharing ("URL password links") is implemented.
- News: the "Refresh now" function is implemented.
- News: the Read/Unread functionality is implemented.
- News: the Feed Delete operation now presents a warning dialog.
- Portals: Single Sign-on operations (to skip the login screen) are implemented.
- Portals: Secure Single Sign-on (SessionID login) is implemented.
- The "screen shot" image on the login screen is customizable now.
- The Files and My Stuff tabs are removed when the Account WebSite service is disabled.
- Telephony features (dialer, phones in IM and contacts, Telephony tab, related settings pages, related context menu items) are removed when the Account PBX service is disabled.
- Dialer: Contact search for incoming phone numbers is implemented.
- Dialer: the Redial button has been added.
- Dialer: new attended transfer function: press the 'transfer button' for 3 seconds.
- Dialer: new attended transfer function: drag the avatar between the phone lines.
- Calendar: grid row size for 15-minute Calendar slices is increased.
- Calendar: the Event owner is now detected on the server side.
- My Stuff and its sub-sections are re-implemented as separate modules.
- My Stuff: (AIR only): the "Display/play local files" for music, photo files.
- My Stuff: Downloading operations added for MyStuff modules: Photo and Music.
- Music: playlists management implemented.
- Mail: the AIR version prefers text/html parts over text/plain message parts.
- Mail: HTML parts not associated with the Show HTML button are added to the attachment list.
- Shared folders can be deleted now.
- The Shared folders pop-up menu now includes the 'Unsubscribe' item.
- Mailbox and File "tree views" use dynamic data retrieval now.
- Bug Fix: all available shared folders (as opposed to subscribed folders) were shown in the Mailbox tree.
- Bug Fix: the '&' character was not escaped in text/plain messages.
- Bug Fix: NotificationPopUpButton closing resulted in null object references.
- Bug Fix: the DTMF setting could be reset to default.
- Bug Fix: the bye command was not sent on exit.
- Bug Fix: the "Show HTML" button was displayed only if multipart/alternative part had an HTML subpart.
- Bug Fix: an Email message body could be fetched multiple times in fast-click scenarios.
- Bug Fix: Dialer: Call History item drag-n-drop operations could result in a program exception.
- Bug Fix: if calendar service was not available, the Calendar was still opened at startup.
- Bug Fix: the Conference Manager used an incorrect URI for its "Add Me" button.
- Bug Fix: the Preferences values were ignored when displaying icons for "special" folders (trash, sent items, drafts).
- Bug Fix: the Login screen was not displayed if the 'sid' parameter was passed without the 'username' parameter.
- Bug Fix: the Create Folder operation did not work for shared Mailboxes.
- Bug Fix: when composing messages, the recipient "real name" was always looked for in the Address Book, ignoring the explicitly specified one.
- Bug Fix: Email grid selection jumped to the first element after sending a reply for which a draft has been saved.

- Bug Fix: if Calendar slice was set to 15 or 30 minutes, the grid labels were set incorrectly.
- Bug Fix: Calendar name in the title bar could be displayed incorrectly.
- Bug Fix: Dragging E-mail messages to folders did not work.

**2.1** CommuniGate Pro 5.2.2

- Palettes are implemented.
- ACL management for mail, calendar, contacts Mailboxes is implemented.
- Shared Mailbox subscription management is implemented.
- HD (H.264) Video playback and High Efficiency AAC (HE-AAC) audio support is implemented.
- Folders with names longer then 15 characters can be created now.
- Junk E-mail Rules management is implemented.
- Simplified Signal Rules management is implemented.
- MS Gothic and MS Mincho font support is implemented.
- Experimental: Banner system support is implemented.

**2.0** CommuniGate Pro 5.2.1

- Adobe Air (desktop Pronto!) version is released: drag & drop, independent windows, etc.
- Spell Checker functionality is improved.
- Directory address books are supported now.
- The Conference module is included now.
- File Manager is included.

---

# PBX: Overview

- **PBX Features**

The legacy circuit-switching telephony implemented certain services using "phone switching stations", often called *PBX* (Private Branch Exchange) or Centrex (if the switching equipment was hosted with the telephony provider).

This section lists the features and services often found in legacy PBX solutions, and it explains how these features work in the CommuniGate Pro environment (also known as IP PBX).

The CommuniGate Pro PBX functionality is completely customizable. Each installation can have a different set of "PBX applications", with completely different set of services provided. This section describes the "stock" applications included into the CommuniGate Pro software package.

**Note:** If the CommuniGate Pro Server PBX functionality is used in the trial mode, a reminder message is periodically played for all media sessions terminated within CommuniGate Pro.

If the PBX Service is enabled for your Account, you can use the WebUser Interface to manage your PBX service settings.

---

# PBX Features

The following section specifies the most popular PBX functions and explains how these functions are implemented in the CommuniGate Pro environment.

Auto Attendant (AA)
An AA system allows callers to be automatically transferred to a user's extension without the intervention of a receptionist.
When the CommuniGate Pro System is installed, it creates an Account named `pbx`, assigns it the `200` alias, and assign it a Signal Rule to start the stock `PBX` application. This application implements the Auto-Attendant functions. See the PBX Center section for more details.

Automatic Call Distributor (ACD)
An ACD system distributes incoming calls to a specific group of agents.
The CommuniGate Pro software comes with a pre-designed basic call queue control applications.

Automated Directory Assistance (ADA)
An ADA system allows callers to route calls to given employees by keying the letters of the employee's name.
The CommuniGate Pro PBX Center application includes the Directory Service as an option.

Automatic Ring Back (ARB)

    An ARB system provides callers with an option to be called when the destination they failed to reach becomes available.

Call Accounting

    A Call Accounting system collects data when a call is made and attaches a cost and a location to the call.

    The CommuniGate Pro Signal component generates accounting CDR records. It also maintains the incoming and outgoing call logs in the Account File Storage.

Call Forwarding

    A Call Forwarding system allows an incoming call to a called party, which would be otherwise unavailable, to be redirected to some other telephone number where the desired called party is situated.

    The CommuniGate Pro Call Control features and Signal Rules provide a very powerful and flexible call control environment. It can be used to implement many different types of call forwarding.

Call Park

    The Call Park feature allows a person to put a call on hold at one telephone set and continue the conversation using some other telephone set.

    The CommuniGate Pro Services application implements a multi-line, queue-type Parking Service for each Account. Additionally, the Park Center feature allows Domain users to park a call in a Domain-wide (or System-wide) Park Center, where it can be picked up by any other user of that Domain.

Call Pick-up

    The Call Pick-up feature allows a person to answer someone else's call.

Call Through

    A Call Through feature allows a user to call the PBX system first, and then make the system place a (usually expensive) call on the user's behalf.

    The CommuniGate Pro Services application implements the Call Through feature.

Call Transfer

    A Call Transfer is a mechanism enabling a user to relocate an existing call to another device or extension.

    The CommuniGate Pro fully supports SIP operations required to implement Call Transfer.

    The CommuniGate Pro XIMSS module allows XIMSS-based clients to implement all types of Call Transfer operations.

    The CommuniGate Pro Real-Time Application environment can automatically accept Call Transfer operations allowing calls established with applications (such as IVR) to be transferred without limitations. Real-Time Applications can also initiate all types of Call Transfer operations.

    The CommuniGate Pro PSTN gateway applications implement Call Transfer internally, thus providing unlimited call transfer functionality for PSTN calls.

Call Waiting

    The Call Waiting feature allows a calling party to place a call to a called party which is otherwise engaged, so the called party is able to suspend the current telephone call and switch to the new incoming call, and then back to the previous call.

    The CommuniGate Pro SIP and XIMSS based clients can handle several calls at once, so they either do not need the Call Waiting feature or implement it locally.

    A special CommuniGate Pro Real-Time application can be used to emulate the legacy call waiting feature for users with single-line devices.

Call Return / Camping

    The Call Return feature allows a called party to place a call back to the calling party of the last received call.

    The CommuniGate Pro Services application implements the Call Return feature.

Conference Call

The Conference call is a call with more than two participants.

The CommuniGate Pro Conference application is used to set Conference calls.

Custom Greetings

The Custom Greeting feature allows users to change their announcements according to special criteria.

The CommuniGate Pro Voice Mail application can play Custom Greetings. The CommuniGate Pro Services application can be used to edit Custom Greetings. The Custom greetings can also be uploaded directly to the Account File Storage.

Direct Inward Dialing (DID)

The Direct Inward Dialing feature (offered by telephone companies) allows the customer PBX to receive calls for a range of numbers, to get the information about the number dialed, and to route the call to proper extension based on that information.

CommuniGate Pro receives PSTN calls via PSTN-to-SIP gateways.

PSTN gateways providing "SIP trunking" deliver the DID information as part of their SIP requests. This information can be used directly with the CommuniGate Pro Router to send a call to any local or remote Account, Group, or Application.

Consumer-type PSTN gateways require separate "registration" for each DID. The CommuniGate Pro Remote SIP Registration feature can be used to receive calls from these gateways, and to ensure that each call request contains the necessary DID information.

Direct Inward System Access (DISA)

The Direct Inward System Access feature allows a calling party to access internal features from an outside telephone line.

The CommuniGate Pro PBX Center and Voice Mail applications provide the DISA features.

Extension Dialing

The Extension Dialing feature allows the system users to call each other using short (2-5 digits) numbers.

CommuniGate Pro Accounts can have short numeric Aliases, implementing independent Extension Dialing plans within each Domain.

Follow-me / Find-me

The Follow-me feature routes incoming calls to a person trying each number in a pre-configured list until the call is answered.

The CommuniGate Pro Call Control features and Signal Rules implement very flexible Follow-me configurations. Unlike many other PBX systems, CommuniGate Pro can "fork" calls, adding new numbers/addresses to the set of devices being alerted, with all devices in the set ringing simultaneously.

Message Waiting Indicator (MWI)

The Message Waiting Indicator is an audio or visual signal the telephone device send to inform that a voicemail message is waiting.

The CommuniGate Pro Signal component provide the `message-summary` package. It can be used with any SIP-based device to implement Message Waiting Indicator.

Music on Hold (MOH)

The Music on Hold feature allows the system to play pre-recorded music to fill the silence that would be heard by telephone callers that have been placed on hold.

The CommuniGate Pro Park Center and many other applications use the system ability to play any pre-recorded media file to implement Music on Hold feature.

Night Service

The Night Service feature allows the system to route incoming calls depending on the current time of day.

The CommuniGate Pro Call Control features and Signal Rules implement very flexible Night Service configurations: the calls can be processed depending on the time of day, the working hours of the callee, the callee presences, etc.

Vertical Service Codes (VSC)

The Vertical Service Codes are special short telephone numbers starting with the star (*) symbol/key, used to access system services.

The CommuniGate Pro Services application implements various Service Codes.

Voicemail (voice mail, vmail, or VMS)

Voicemail is a system allowing callers to record and store phone messages, and allowing its users to play and distribute stored messages.

The CommuniGate Pro Voice Mail application allows caller to store messages. Voice messages are delivered to the callee INBOX as any other E-mail message, and can be filtered, sorted, and processed using the Queue Rules.

The CommuniGate Pro Services application can be used to retrieve and manage recorded messages. At the same time, any POP, IMAP, MAPI, XIMSS client or WebUser Interface can be used to retrieve and manage recorded messages.

**Overview** **Call Control** **Voicemail** **Services** **PBX Center** **Conference**

# PBX: Call Control

- **Addresses**
- **Launching Voice Mail**
- **Simultaneous Ringing**
- **Call Diverting**
- **Call Blocking**
- **Call Logs**

The CommuniGate Pro [Signal](#) component can use [Automated Rules](#) to process calls and other Signals.

If the Administrator granted you a proper right, you can create and modify Signal Processing Rules for your Account.

To simplify this process, the [WebUser Interface](#) and [XIMSS](#) clients allow you to set so-called Simplified Rules. The Simplified Rules have predefined conditions and actions, and you simply enable or disable them, and provide some basic parameters.

This section explains how you can set your Simplified Signal Rules using the "stock" WebUser Interface. To set these Rules, log into your Account using the WebUser Interface and open the Call Control page.

# Addresses

You need to specify "addresses" to tell your Automated Rules where to send the call to. You can specify an address as:

- a simple name or a number (in this case, this address will be interpreted as an address in your Domain),
- a fully-qualified addresses (`name@domainName`),
- a complete SIP URI (such as `<sip:name@domainName;parameters>`)

You can specify several addresses if you separate them using the comma (`,`) symbols.

You can direct a call to an application, using the application name prefixed with the hash (`#`) symbol as an address (`#applicationName`).

# Launching Voice Mail

The following section specifies the most popular PBX functions and explains how these functions are implemented in the CommuniGate Pro environment.

| Voice Mail | | |
|---|---|---|
| After:　20 sec | On Busy | On Error |

**After**
> If this option is selected, then the voicemail application will be started after the specified period of time, or immediately, if your Account has no registered device.

**On Busy**
> If this option is selected, then the voicemail application will be started if any of your devices return the "busy" (486) or "busy everywhere/DND" (600) or "Declined" (603) code.

**On Error**
> If this option is selected, then the voicemail application will be started if the call fails for any reason other than "no answer", "no response", "busy", "busy everywhere", "declined".

See the section to learn how the Voice Mail application works.

# Simultaneous Ringing

You can configure your Account to ring additional devices, persons, or applications.

| Simultaneous Ringing | |
|---|---|
| After:　5 sec | Fork to: |

**Simultaneous Ringing**
> When this option is enabled, the incoming calls causes your registered devices to ring. If you do not answer for the specified period of time, the call is "forked" to some other address (device, person, application), while your devices continue to ring. The call is connected to any device that answers it, and at this moment calls to all other devices are canceled.

**After**
> Use this option to specify for how long the system will ring your own devices before it starts to "fork" to other addresses.

**Fork to:**
> Specify one or several phone numbers or E-mail addresses to direct the incoming call to.

You can use this feature to "fork" your calls to your mobile phone, to your assistant, etc.

If your Account has no registered device, and this option is enabled, the call is immediately "forked" to the specified addresses.

# Call Diverting

You can configure your Account to redirect or reject incoming calls based on certain conditions.

| Divert Calls | | | |
|---|---|---|---|
| **When:** | always<br>afterhours<br>work hours | **To:** | voice mail<br><br>busy signal |

|  | - |  |  |
|--|---|--|--|

**When**

Use this setting to specify when Call Diversion should take place. To learn if the current time is your "afterhours" or "work hours" time, CommuniGate Pro uses your [Calendar](#) Settings.

**To**

Use this setting to specify where incoming calls should be diverted to:

- voicemail - to the [Voice Mail application](#)
- address - to the specified [address](#)
- busy signal - the signal is rejected with this Rule.

# Call Blocking

You can configure your Account to block certain call from ringing your devices.

| **Block Calls** | | | |
|---|---|---|---|
| **From:** | Blacklisted Addresses | **To:** | voice mail<br>busy signal |

**From**

Use this setting to specify which calls should be blocked:

- Blacklisted - the addresses included into the [Blocked](#) Address Book.

**To**
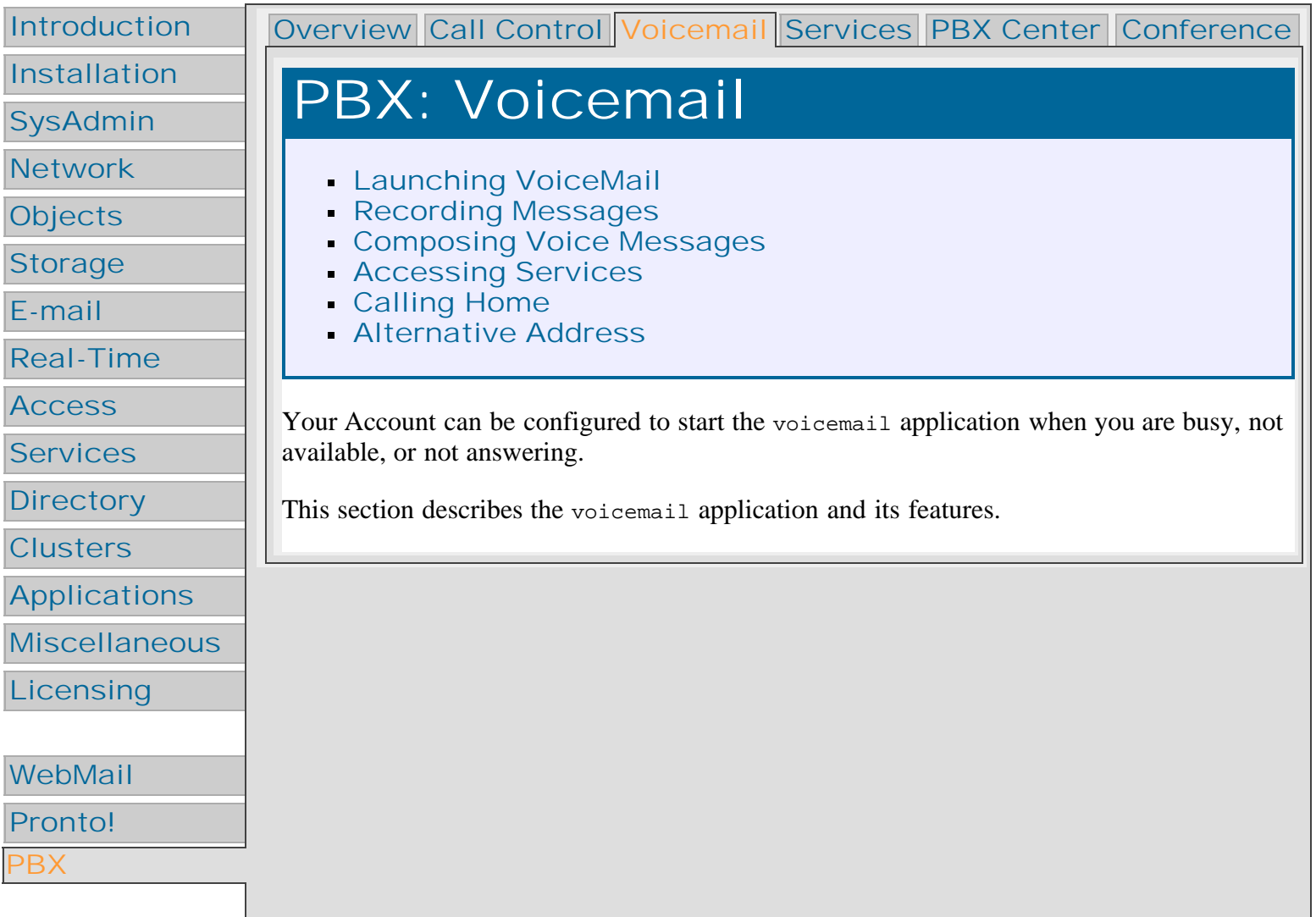
Use this setting to specify how to block the calls:

- voicemail - redirect calls to the [Voice Mail application](#)
- busy signal - reject calls

# Call Logs

CommuniGate Pro places information about your Incoming and outgoing calls in the Call Logs, and stores them as files in your Account [File Storage](#).

You can use the [WebUser Interface](#) to view these logs:

| Dec 2007 | | Filter: | | | 5 of 189 selected | |
|---|---|---|---|---|---|---|
| **When** | **Type** | **Name** | **Party** | **Time** | **Cost** |  |
| 03-Dec 3:22:30AM | incoming | John Smith | john@smith.dom |  |  |  |
| 07-Dec 1:01:17AM | incoming |  | +1415551212 |  |  |  |
| 07-Dec 3:03:42PM | outgoing |  | 275 |  |  |  |
| 08-Dec 7:36:33PM | incoming | CG Sales | sales@communigate.com |  |  |  |
| 09-Dec 7:28:12PM | outgoing |  | 16505551212 |  |  |  |

Overview **Call Control** **Voicemail** Services PBX Center Conference

# PBX: Voicemail

- **Launching VoiceMail**
- **Recording Messages**
- **Composing Voice Messages**
- **Accessing Services**
- **Calling Home**
- **Alternative Address**

Your Account can be configured to start the `voicemail` application when you are busy, not available, or not answering.

This section describes the `voicemail` application and its features.

# Launching VoiceMail

To specify when the `voicemail` application should accept your incoming calls, open the Call Control WebUser Interface page.

The voicemail application can be configured to start when:

- your Account has no registered device
- none of your devices is picked up after certain "ringing" (alerting) time
- all your devices have rejected the call
- you have pressed the DND (Do Not Disturb) button on any of your registered devices
- the caller is the Blocked list
- the current time is outside your working hours
- your current Presence status is set to the pre-defined value (busy, away, etc.)

A Router Default Record starts an Account `voicemail` application when a \**nnn* number is dialed, where *nnn* is an Account numeric alias.
This feature can be used by a receptionist to transfer or redirect certain calls directly to the callee's voicemail.

# Recording Messages

The stock `voicemail` application performs the following operations:

- The application plays a greeting. It can be the standard greeting or your custom greeting.
- The application allows a caller to record a voice message, to review that message, to re-record the message, to mark the message as urgent, or to cancel the recorded message.
- The application allows a caller to listen to custom greeting (private announcements).
  The caller should press `6` and provide the correct [announcement PIN](#).
- The application allows a caller to join your personal conference. The caller should press `7` and provide your personal Conference PIN that you have set using the [WebUser Interface](#).

---

# Composing Voice Messages

Voice Messages are composed as voice-type E-mails and mailed to your Account.

Your Account Queue/Mail [Rules](#) will be applied to the voice messages before they are stored in your INBOX.
If you E-mail client does not support voice-type E-mails, you will see voice messages as audio-file attachments. You can play these messages by opening these attachments.

---

# Accessing Services

You can use the `voicemail` application to start your [service application](#) when you cannot call from your own VoIP phone (for example, when you are calling from a PSTN phone via a gateway).

Press the `*` (star) symbol twice while listening to the greeting and/or the menu options. The application will ask you to enter your Access PIN. If the number is correct, the application switches you to the [service application](#) menu.

---

# Calling Home

You can use the [WebUser Interface](#) to configure your Home PSTN Number.

The voicemail application compares the local part of the caller address (the phone number) with the Home PSTN Number value. If the tail of the address matches the specified value, the application asks the caller to enter the Access PIN and then it switches to the [service application](#).

Since the application checks only the address tail, you can specify your Home PSTN Number as a local number, or as a number without the country code.

---

# Alternative Number

You can use the [WebUser Interface](#) to configure the Alternative Number. If callers press the `9` button, they are transferred to the Alternative Number.
For example, you can specify the `pbx` or `operator` address with this setting, so the callers can transfer themselves to the company automatic or live attendant if they do not want to leave a voicemail for you.

---

Introduction

Installation

SysAdmin

Network

Objects

Storage

E-mail

Real-Time

Access

Services

Directory

Clusters

Applications

Miscellaneous

Licensing

WebMail

Pronto!

PBX

| Overview | Call Control | Voicemail | **Services** | PBX Center | Conference |

# PBX Services

- **Accessing Services**
- **Access PINs**
- **Mailbox Management**
- **VoiceMail Greeting**
- **Custom Greeting**
- **Call Park and Parked Call Pick up**
- **Missed Call Pick up**
- **Personal Conferencing**
- **Last Call Return**
- **Dialing Out**
- **Speed Dial**
- **Diverting Calls**
- **Blocking Call ID**
- **Incoming Call Pick up**

A special `service` application is started when you "dial your own number" (i.e. make a call to your own E-mail address), or when you dial a `*NN` number, where *NN* is any 2-digit number.

This section describes the telephony services implemented with the stock `service` application.

## Accessing Services

The Signal component automatically stats the `service` application when you call your own account. The same application is started if you dial any `*NN` number, where *NN* is any 2-digit number.

The `service` application checks how it was invoked. If it was launched to serve a `*NN` call, it uses *NN* as the function code. Otherwise, it presents the functions menu and asks you to enter the function code.

You can invoke the `service` application by calling your Account from any device and using the `voicemail` application Access PIN feature.

You can invoke the `service` application by using the `Auto-Attendant` application Service Access feature.

## Access PINs

Access PINs (numeric passwords) are required to access the `service` application from the `voicemail` application and/or `auto-attendant` application.
A Conference PIN is required to access your Personal Conference.

Use the WebUser Interface or a XIMSS client to set these PINs:

| PINs (Numeric Passwords) |
| --- |
| Service Access PIN: |
| Conference PIN: |

Service Access PIN
> Enter a number into this field. You will use this number (PIN) to access your Account <u>services</u> when connecting to the system from devices not registered under your name.

Conference PIN
> Enter a number into this field. You will provide this number (PIN) to the individuals who should have access to your <u>Personal Conferencing</u> facility.

---

# Mailbox Management

The <u>service application</u> allows you to check the content of your Account Mailboxes (folders).
By default, the application provides access to your Inbox Mailbox.

You can bypass the mail service menu and connect to you Inbox Mailbox directly by dialing `*51`.

Follow the Mailbox Management menu to listen to your messages, to mark your messages as read, to mark your messages for deletion, and to forward your messages to other extensions.

---

# Voicemail Greeting

The <u>voicemail application</u> checks if your Account <u>File Storage</u> contains a `mailprompt.wav` file. If this file can be read, it is used as your voicemail greeting, otherwise the standard greeting is played.

You can create, modify, or remove your personal greeting using any of the <u>File Storage</u> modification methods - FTP, WebUser Interface, etc.

You can create, modify, or remove your personal greeting by starting your <u>service application</u> and selecting the Greeting Modification option.

---

# Custom Greeting

You may want to create custom greetings in your Account to leave personal messages or announcements for certain callers. Each custom greeting is associated with a PIN, and a caller needs to know the greeting PIN to listen to the custom greeting.

Dial `*53` or connect to your <u>service application</u> and enter the `53` code to edit your custom greetings.
The application will ask you to enter a PIN - select any number you want (up to 20 digits). Then the application will allow you to record, rewrite or remove the custom greeting associated with that PIN.

Custom Greetings are stored in your <u>File Storage</u> as `private/greetings/`*NNNNNNN*`.wav` files, where *NNNNNNN* is the greeting PIN.

You can also create, modify, or remove your custom greetings using any of the [File Storage](#) modification methods - FTP, WebUser Interface, etc.

# Call Park and Parked Call Pick up

You may want to "park" a current call by disconnecting from the peer without breaking the call. Later you can reconnect to the peer (to "pick up" the call), using the same or a different phone or device.

To park a call, use the phone "blind transfer" function to transfer the peer to the `*55` number. The peer will be connected to a service application playing music-on-hold, and your device will be disconnected.

To pick up a call, dial the `*57` number from any of your own devices. The `service` application finds your parked call and connects you to the peer.

You may want to use any device to pick up your parked call. Dial your own number to connect to your `voicemail` application and use the [Access PIN](#) feature to access your `service` application.
Then enter the `57` code to pick up the call.

You can park several calls at the same time. When you pick up the parked calls, they are retrieved in the same order they were parked.

You may want not to send music-on-hold to a parked call (for example, when you park a conference call with active participants). Blind-transfer a call to the the `*56` number for "silent parking".

If you dial the `*55` or `*56` number, your own call will be parked.
You can use your other device to pick up this call, establishing a call between your own devices.

A "parked peer" can press the `#` button to disconnect.

# Missed Call Pick up

If you have just missed a call and the call is being answered with the [voicemail application](#), you can pick up that call while the caller is leaving you a message.

Dial `*59` or connect to your [service application](#) and enter the `59` code to pick up a missed call.

# Personal Conferencing

You can use your CommuniGate Pro Account for multi-party conferencing.

Use the [WebUser Interface](#) to set the Conference PIN. Send this PIN to the parties you want to have a conference with.

Dial `*61` or connect to your [service application](#) and enter the `61` code to activate your personal conference. Now you are the [Conference Host](#).

Other participants can dial into your Account to connect to your [voicemail application](#) and select the Conference option. If they enter the correct Conference PIN, they will be joining your conference.

If you try to start your conference when it is already active (you have started it using some other device), you will be

joining the conference as a participant.

Dial `*65` or connect to your <u>service application</u> and enter the `65` code to "clear" your personal conference. You can now start a new conference, even if there is an active conference in progress.

# Last Call Return

You can return the last call received by your Account.

Dial `*69` and the service application will retrieve the information about the last incoming call, and it will redirect your device to the originator of that call.

# Dialing Out

You can ask the system to make a call for you.

Dial `*40` or connect to your <u>service application</u> and enter the `40` code, and enter the number you want to call. The system will dial that number on your behalf (on behalf of your Account).

# Speed Dial

You can use the system to help you quickly dial the most often used addresses and/or phone numbers.
Use the <u>WebUser Interface</u> to specify up to 9 speed-dial numbers. Each number is associated with the speed-dial digit code (`1`, `2`, ... `9`).

Dial `*4N` or connect to your <u>service application</u> and enter the `4N` code, where *N* is the speed-dial digit code. The system will dial that number on behalf of your Account.

# Diverting Calls

Use the <u>WebUser Interface</u> to specify a simplified "Divert" Rule, or you can use your phone to create or remove this Rule:
Dial `*70` or connect to your <u>service application</u> and enter the `70` code. Enter the phone number or a numeric alias to redirect all your calls to (without any time condition). The "Divert" Rule will be created and enabled.
Enter an empty number (press `#` without digits) to remove the "Divert" Rule.

You can enable or disable the Divert Rule using your phone (this Rule must already exist):
Dial `*71` or connect to your <u>service application</u> and enter the `71` code enable the Divert Rule.
Dial `*72` or enter the enter the `72` code to disable the Divert Rule.

# Blocking Call ID

When making calls to PSTN, you may want to block your system from sending your phone number information to the called party.
Use the <u>WebUser Interface</u> to control the Call ID Blocking setting.

You can also control that setting using your phone:

Dial *77 or connect to your service application and enter the 77 code to block Call ID sending.

Dial *78 or enter the enter the 78 code to unblock Call ID sending.

---

# Incoming Call Pick up

The system can be configured to allow you to answer incoming calls directed to other users. When you hear an incoming call alerting for a different user (or you detect such a call via other means), you can dial a special number to answer that call.

By default, the Router records direct all calls made to 8*nnn* to the pickup application.
That application intercepts the incoming call pending for the user whose Account alias is *nnn*. I.e. to pick up an incoming call for the user Account j.smith whose numeric alias is 245, you need to dial 8245.

You can pick up incoming calls directed to a different user only if that user has granted you the Call Control access right, or if you have the CanControlCalls Domain Access right.

---

Overview | Call Control | Voicemail | Services | **PBX Center** | Conference

# PBX Center

- **The PBX Center Account**
- **The PBX Application**
- **Home Calls**
- **Auto-Attendant**
  - **Hierarchical Menus**
  - **Service Access**
  - **Voicemail Access**
  - **Personal Conference Access**
- **Conference Center**
- **Call Park Center**

A PBX Center is a special [Account](#) in your CommuniGate Pro [Domain](#). This Account uses a special `pbx` Real-Time Application to answer calls from other Accounts in the same Domain, as well as calls from "external users" (Accounts in other Domains, accounts on other servers, PSTN users calling via gateways, etc.)

A PBX Center allows users of your Domain to collaborate on call processing.

## The PBX Center Account

When the CommuniGate Pro Server software is installed, it automatically creates a PBX Center Account `pbx` in the Main Domain, and it creates the `200`, and `conference` [Aliases](#) for that Account.

Remote VoIP users can access the PBX Center by calling `sip:pbx@your.domain.name`, and users with Accounts in your Domain can access your PBX Center by dialing `200` on their VoIP phones.

If your Domain uses PSTN gateway(s) for incoming calls, you may want to configure those gateways to direct all incoming calls to your PBX Center Account.

You can also use the PBX Center Account address as the "public VoIP address" of your organization.

If you are a Server or a Domain Administrator and you want to create an alternative PBX Center:

- Create an [Account](#) to be used as a PBX Center.
- Change the Account Real-Time settings so the `pbx` application is started immediately when a call comes in.
- Grant the [CanImpersonate](#) Domain Access Right to the PBX Center Account, so the application can place calls on behalf of your Domain users.

# The PBX Application

When the PBX Center is called directly (using the `pbx` name or the `200` Alias), it starts the <u>Auto-Attendant application</u>.

Other PBX Center functions can be invoked by accessing the PBX Center via alternative names. By default, the CommuniGate Pro [Router](#) directs all calls to `7nn` names to the `pbx` Account (where *nn* is a 2-digit number).

When the PBX Center application starts, it checks the address used to call it. If this address (the "local part" of it, the part before the domain name) is a 3-, 4-, or 5-digit number, the application uses the last 2 digits as the requested function number. The documentation below assumes that your Domain uses the `7nn@domainName` address to invoke the PBX Center function number *nn*.

The PBX Center application requires authentication for most calls coming from its own Domain. You may need some exceptions from this rule. For example, your in-house gateway may send incoming calls to your PBX Center using the `From:<sip:gate1@mydomain.com>` address.
To accept these request without authentication, create the `ExternalGateways` [Group](#) in your Domain and include the `gate1` name there.
The `gate1@mydomain.com` must be routable: for example, you may want to create a `gate1` Account, or, more likely a `gate1` Alias for one of your existing Accounts.
If a request is not authenticated, the PBX Center processes it as an "external call".

---

# Home Calls

The PBX Center application can recognize calls from certain PSTN numbers as "local users" calls.
If you call the PBX Center application (usually - the main company PSTN number) from your cell or other PSTN phone, you are asked for your Access PIN. When the PIN is accepted, you are connected to your <u>Service application</u>.

To make the PBX Application recognize your "home calls", your Account should have a `homecall-pstnNumber` alias, where *pstnNumber* is your PSTN number (digits only). If your number is longer than 10 digits, use only the last 10 digits of it.

---

# Auto-Attendant

The `Auto-Attendant` application answers incoming calls and presents a menu. Callers use this menu to select a Domain Account or a service to connect to. The application dials the selected address, and if a connection is established, the application connects ("bridges") the incoming call and the called party.

The Server or a Domain Administrator should configure the `pbx` application Preferences. Use the WebAdmin Interface to open the Real-Time Setting pages of the PBX Center Account, and follow the Advanced link to open the `Auto-Attendant` application Preferences:

| Auto-Attendant |
|---|
| Language Menu: |
| Department Menu: |
| Directory Prefix: |
| Directory Digits: |

Language Menu

> If this list is not empty, the `reception` application presents a language menu. The default Language is specified in the WebUser Preferences of this PBX Center Account. If a list is not empty, the default language is always presented as the first option (even if it is not included into the list).
>
> **Note:** before adding any Language option, make sure that the language has been added to the [Real-Time Application Environment](#).

Department Menu

> Use this list to specify the "departments" the caller can connect to. For each "department name" used, your Domain should contain an Account or other object (Alias, Group, Forwarder) with that name.
>
> For each "department name" used, the [Real-Time Application Environment](#) of your Domain should contain a media file for`name`.wav, where *name* is the "department name".
>
> The stock Environment contains media files for the following "department names": `administration`, `conference`, `engineering`, `frontdesk`, `helpdesk`, `marketing`, `operations`, `sales`, `techsupport`.
>
> If the list contains the `operator` name, the operator option is always listed as the last one, and it is always assigned the `0` ("zero") menu option.

Directory Prefix, Directory Digits

> The `reception` application can connect callers with any Account in your Domain. To facilitate calling from devices that have a digital keypad only, Accounts should have digital [Aliases](#) (called "extensions" in legacy PBX systems).
>
> It is recommended to use the same number of digits in each Alias, and start all Aliases with the same digit.
>
> If Accounts in your Domain have 3-digit Aliases, and all of them start with the digit `2`, specify this digit as the Directory Prefix, and specify `3` as the Directory Digits setting.
>
> If the Directory Prefix digit is specified, the `reception` application does not use that digit as a menu option. If the caller enters that digit, the application waits till the complete number is entered (i.e. as many digits as specified with the Directory Digits setting), and then it tries to call that number by calling the address entered.
>
> If the Account `john@domain1.dom` has `202` as its Alias, then anyone can call this Account by connecting to the `reception` application in the `domain1.dom` Domain and entering 202.
>
> If you need more than one Directory Prefix, enter them as one string. For example, if your numeric Aliases ("extensions") are *2xx* and *4xx*, enter `24` as the Directory Prefix value.

If the PBX Center Account has a `receptionprompt.wav` file in its [File Storage](#), then this file is played instead of the standard "Welcome" file.
If the PBX Center Account has a `receptiontrailer.wav` file in its [File Storage](#), then this file is played after the menu options are played.

## Hierarchical Menus

Before dialing the specified destination, the application checks if the destination is an Account on the same system, and if that Account has a non-empty Department Menu preference. If it does, the Account is not called, but its Menu is presented.

For example, when `3` is dialed, the PBX Center is configured to call the `techsupport` Account. If this Account has a Department Menu preference containing `productXsupport` and `productYsupport` items, then this "support submenu" is played (the `forproductXsupport.wav forproductYsupport.wav` should be placed into the PBX Environment, and the `productXsupport` and `productYsupport` should be valid, "callable" addresses).

## Service Access

You can use the [Auto-Attendant application](#) to access your [Service application](#).

Dial your own extension number, prefixed with `*`. The application will ask you to enter your Access PIN. If the number is correct, you are switched to the `Service application` menu.

## Voicemail Access

A caller can use the `Auto-Attendant application` to connect directly to someone's voicemail, by dialing `*NNN`, where *NNN* is the target Account extension (numeric Alias).

## Personal Conference Access

A caller can use the `Auto-Attendant application` to connect directly to the Personal Conferencing facility of some Account, by dialing:
`**NNN#`, where *NNN* is the Account extension (numeric Alias),
or by dialing
`**NNNPPPP#` where *PPPP* is the Account Conference PIN.

---

# Conference Center

The PBX Center application includes the Conference Center functionality.

To connect to the Conference Center, use the PBX Center function `60` by dialing `760`.

Follow the menu to create a Conference. The application will tell you the PIN for the newly created Conference and it will E-mail this PIN to your CommuniGate Pro Account.

If you create an "open" conference, then anyone who knows the Conference PIN can start the Conference. Otherwise, the E-mail sent to you will contain the Leader PIN, and only you can start this Conference.

To remove a Conference, connect to the Conference Center and use its main menu to select the remove function.

To start a Conference, or to join a Conference created by someone else, connect to the Conference Center. Use its main menu to select the "join conference" function, then provide the Conference PIN.
If you are the person who has created the selected Conference, the Conference will be started and you will become the Conference Host. Otherwise you will join the Conference as a participant.

If the PBX Center Account is called using any address starting with the symbols `conference`, the Conference Center function is started.
The `conference` Alias of your `pbx` Account:

- allows remote users to connect to your Conference Center by using the `sip:conference@your.domain.name` address.
- allows PSTN callers to connect to your Conference Center by dialing in, connecting to your PBX Center `Auto-Attendant application`, and selecting the "join a conference" option.

When your Conference Center receives a call from a user outside your Domain, the Center does not present its main menu: it immediately tells the caller to enter the Conference PIN. The caller than joins the conference as a participant, or starts it and becomes the Conference Host, if this is an "open" Conference, and it has not been started yet.

You may need to start your Conference when your call cannot be authenticated (for example, you are calling into your system via a PSTN gateway).
The Conference Center will ask you for the Conference PIN.
Enter the Conference PIN and the application will inform you that the Conference has not been started yet.

Press the star (*) key and enter your Leader PIN.
The Conference will be started and you will become its [Conference Host](#).

The PBX Center function `61` allows you to bypass the main menu: dial `761` and proceed by entering the Conference PIN.

---

# Call Park Center

The PBX Center application includes the Call Part Center functionality.

You may want to "park" a call, so your peer will be connected to a music-on-hold application, while your device will be disconnected from the call. Then you or some other user can "pick" the "parked" call, i.e. to connect to the peer of the initial call.

The PBX Center application offers an unlimited number of "parking queues", and it provides quick access to the first 9 of them.

To park a call in the queue number *n*, connect the peer to your PBX Center function `7n` by transferring the call to the `77n` address.

You and other users of your Domain can park multiple calls in the same queue. The calls will be picked up in the same order as they were parked.

To pick up a parked call from the queue number *n*, you or other user of your Domain should connect to your PBX Center function `8n` by dialing the `78n` address.

---

# Conference

- **Floor Control**

The CommuniGate Pro Media Server can merge media from several "legs", creating a single "conversation space". This feature is used to implement various Conferencing services.

This section explains how a conference "host" can start and control the conference, and how the conference participants are served.

# Floor Control

When you are a Conference Host, you can manage the conference "floor".

Enter `*5` to play the active participants list. This list is played to everyone in your conference.

If you are the only participant in the conference, the system plays music-on-hold. It stops playing it as soon as any other participant joins the conference, and starts playing it again as soon as you are left alone.
Enter `*8` to explicitly start or to stop music-on-hold. If there are several participants in the conference, all of them hear the same music-on-hold.

Enter `*9` to enable or disable "reminder beeps". Reminder beeps are sent to all participants every 30 seconds (unless there is music-on-hold playing).

When there are many participants in one conference, their combined background noise may start to interfere with the conversation. The system "cuts" the participants who are not speaking, in order to reduce the background noise.
If your conference has many participants and you hear too much of background noise, you may want to increase the "cutting threshold".
If the "cutting threshold" is too high, you will hear some speaker voices being "cut" at the word/phrase boundaries.
Enter `*1` to increase the "cutting threshold".

Enter `*2` to decrease the "cutting threshold".

The conferencing software introduces a small delay, letting audio data from all participants to be collected before it is being mixed. If some participants use slow or unreliable Internet connections and voice packets from them arrive with large or variable delays, you may want to increase the "mixer delay".
Enter `*3` to increase the "mixer delay".
Enter `*4` to decrease the "mixer delay".

---