
CommuniGate® Pro

*Internet Communications Server
Email ~ Collaboration ~ VoIP*

CommuniGate Pro v5.1



CommuniGate
SYSTEMS

CommuniGate® Pro 5

- Features and Standards 11
- Revision History 35
- How To 51
- Help Me 63

Installation 71

- Quick Start 93
- Migrating to CommuniGate Pro 95

System Administration 113

- Server Logs 135
- Router 145
- Protection 167
- Security 185
- Public Key Infrastructure 199
- Lawful Interception 217
- Scalability 221
- Alerts 239
- Triggers 243

Network 249

- Listeners 259
- Dialup 265

Objects 269

- Domains 277
- Mapping 301
- Accounts 305
- Groups 327
- Forwarders 333
- Mailboxes 337
- File Storage 349
- Account Data 355

Message Transfer 359

- Automated E-Mail Processing Rules 373
- External Filters 391
- SMTP Module 397
- Local Delivery Module 421
- RPOP Module 431
- LIST Module 439
- PIPE Module 473

Real Time Signals 481

- Automated Signal Processing Rules 493
- SIP Module 499
- XMPP Module 565
- PSTN 567

Account Access 575

- Mailbox Sharing 583
- POP Module 587
- IMAP Module 595
- WebUser Interface 603
- XIMSS Module 609
- MAPI Connector 613
- FTP Module 637
- TFTP Module 641
- ACAP Module 645

Services 647

- HTTP Modules 649
- LDAP Module 661
- PWD Module 669
- RADIUS Module 673
- SNMP Agent 677
- BSDLog Module 681

Directory 683

- Directory Schema 701
- Directory Integration 707

Clusters 757

- Static Clusters 771
- Dynamic Clusters 777
- Cluster Storage 787
- Cluster Transfer 831
- Cluster Signals 895
- Cluster Access 905

Applications 921

- Data Formats 923
- Command Line Interface (API) 929
- CommuniGate Programming Language (CG/PL) 983
- Automated Processing (Rules) 1023
- Helper Applications 1033
- Real-Time Application Module 1045
- XIMSS Protocol 1147
- Web Application Module 1177
- WSSP Scripting 1187
- Web Application Code Components 1213

WebUser Interface (WebMail) 1259

- WebUser Interface: Mailboxes 1271
- WebUser Interface: Messages 1281
- WebUser Interface: Composing Messages 1289
- WebUser Interface: Contacts 1297
- WebUser Interface: Calendar 1307
- WebUser Interface: Tasks 1323
- WebUser Interface: Notes 1331
- WebUser Interface: Secure Mail (S/MIME) 1333

PBX Services 1345
Miscellaneous 1359



CommuniGate® Pro

Welcome to CommuniGate Pro, the Internet Communication Server.

Based on the open standards, the product provides an integrated platform for "store-and-forward" (E-mail, Cal-endaring) and Real-Time (VoIP, Video, Instant Messaging, White Boards) communications.

The main Server subsystems include:

Identity Management

- [Multi-Domain](#) architecture (field-proven for over 50,000 domains per system), with multihoming and shared-IP configurations.
- [Account](#) concept, including Account Storage, Account Settings, and Account Information databases.
- [Groups](#), [Forwarders](#), [Aliases](#), and other [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.

Storage Management

- [Mail Store](#) with multiple mailboxes, [shared access](#), [ACLs](#).
- [Mailbox formats](#) - text files, file folders, other data containers.
- [File Store](#) with public and private folders, virtual files.
- [Groupware Information](#) storage and processing using the iCalendar and vCard standards.

Mail Transfer

- [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 Signalling

- [SIP](#) protocol for Instant Messaging, audio, and video communication.
- [XMPP](#) protocol for Instant Messaging and Presence.
- [NAT Traversal](#) mechanisms (near-end and far-end) for RTP and TCP-based media protocols.
- [Registrar](#), forking [Proxy](#), and [Presence](#) server functionality.
- [Automated Signal Processing](#) Rules.
- [Event packages](#) for presence, message-waiting, registration, and other services.

Real-Time Application Environment

- [Domain-specific](#) application environments.
- [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

- [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.
- [Multi-lingual Skins](#) for customizable HTML, WAP/WML, and I-Mode Interfaces.
- [XML API](#) providing access to Mailbox, Groupware and File Stores, to Settings and Information databases, to the Signaling and Message Transfer facilities.
- [HTTP](#), [FTP](#), and [TFTP](#) access to Account File Stores.
- [Publish/Subscribe](#) HTTP-based operations with Calendar and Tasks 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 remotely (via the Internet) using any Web browser. CommuniGate Pro has a built-in HTTP (Web) server so it does not require any additional Web Server application to support remote configuration features, and it does not conflict with any other Web Server running on the same computer.

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://mail.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.

Stalker Software contact and general information is available at <http://www.stalker.com>.

Updates and Bug Fixes

CommuniGate Pro is being updated on a regular basis.

You can review the history of updates and bug fixes at the [CommuniGate Pro History page](#)



Features and Standards

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 industrial-strength messaging systems..

Kernel

Supported Standards

[RFC4021](#)

Registration of Mail and MIME Header Fields.
G. Klyne, J. Palme. March 2005.

[RFC3986](#)

Uniform Resource Identifiers (URI): Generic Syntax.
T. Berners-Lee, R. Fielding, L. Masinter. January 2005.

[RFC3548](#)

The Base16, Base32, and Base64 Data Encodings.
S. Josefsson, Ed.. July 2003.

RFC3330	Special-Use IPv4 Addresses. IANA. September 2002.
RFC2732	Format for Literal IPv6 Addresses in URL's. R. Hinden, B. Carpenter, L. Masinter. December 1999.
RFC2392	Content-ID and Message-ID Uniform Resource Locators. E. Levinson. August 1998.
RFC2387	The MIME Multipart/Related Content-type. E. Levinson. August 1998.
RFC2047	Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header Extensions for Non-ASCII Text. K. Moore. November 1996.
RFC2046	Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. N. Freed & N. Borenstein. November 1996.
RFC2045	Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. N. Freed & N. Borenstein. November 1996.
RFC2044	UTF-8, a transformation format of Unicode and ISO 10646 Yergeau, F. October 1996.
RFC1700/ STD0002	ASSIGNED NUMBERS. J. Reynolds, J. Postel. October 1994.
RFC1123	Requirements for Internet Hosts -- Application and Support R. Braden, Editor. October 1989.

Security

Supported Standards

RFC3961	Encryption and Checksum Specifications for Kerberos 5. K. Raeburn. February 2005.
RFC3174	US Secure Hash Algorithm 1 (SHA1). D. Eastlake, P. Jones. September 2001.
RFC2831	Using Digest Authentication as a SASL Mechanism. P. Leach, C. Newman. May 2000.
RFC2633	S/MIME Version 3 Message Specification. B. Ramsdell. June 1999.
RFC2632	S/MIME Version 3 Certificate Handling. B. Ramsdell. June 1999.
RFC2630	Cryptographic Message Syntax. R. Housley. June 1999.
RFC2617	HTTP Authentication. Basic and Digest Access Authentication 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. C. Newman. June 1999.
RFC2585	Internet X.509 Public Key Infrastructure. Operational Protocols: FTP and HTTP R. Housley, P. Hoffman. May 1999.
RFC2478	The Simple and Protected GSS-API Negotiation Mechanism. E. Baize, D. Pinkas. December 1998.
RFC2459	Internet X.509 Public Key Infrastructure. Certificate and CRL Profile. R. Housley, W. Ford, W. Polk, D. Solo. January 1999.

RFC2315	PKCS #7: Cryptographic Message Syntax. Version 1.5 B. Kaliski. March 1998.
RFC2246	The TLS Protocol. Version 1.0 T. Dierks. C. Allen, January 1999.
RFC2222	Simple Authentication and Security Layer. J. Myers. October 1997.
RFC2195	IMAP/POP AUTHorize extension for Simple Challenge/Response. J. Klensin & others. September 1997.
RFC2078	Generic Security Service Application Program Interface, Version 2 J. Linn. January 1997.
RFC2104	HMAC: Keyed-Hashing for Message Authentication. H. Krawczyk, M. Bellare, R. Canetti. February 1997.
RFC964	The Kerberos Version 5 GSS-API Mechanism. J. Linn. June 1996.
RFC1731	IMAP4 Authentication Mechanisms. J. Myers. December 1994.
RFC1510	The Kerberos Network Authentication Service (V5) J. Kohl, C. Neuman. September 1993.
RFC1321	The MD5 Message-Digest Algorithm. R. Rivest. April 1992.

International

Supported Standards

RFC3629	UTF-8, a transformation format of ISO 10646. F. Yergeau. November 2003.
RFC2319	Ukrainian Character Set KOI8-U. KOI8-U Working Group. April 1998.
RFC2278	IANA Charset Registration Procedures. N. Freed, J. Postel. January 1998.
RFC2184	MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations. N. Freed, K. Moore. August 1997.
RFC2152	UTF-7 A Mail-Safe Transformation Format of Unicode. D. Goldsmith. M. Davis. May 1997.
RFC1557	Korean Character Encoding for Internet Messages. U. Choi. K. Chon,H. Park. December 1993.
RFC1489	Registration of a Cyrillic Character Set. 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

RFC2183	Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field. R. Troost, S. Dorner, K. Moore. August 1997.
RFC2111	Content-ID and Message-ID Uniform Resource Locators. E. Levinson. March 1997.
RFC2076	Common Internet Message Headers. J. Palme. February 1997.
RFC1894/RFC3464	An Extensible Message Format for Delivery Status Notifications. K. Moore & G. Vaudreuil. January 1996.
RFC1892/RFC3462	The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages. G. Vaudreuil. January 1996.
RFC1740	MIME Encapsulation of Macintosh files - MacMIME. P. Faltstrom, D. Crocker, E. Fair. December 1994.
RFC1741	MIME Content Type for BinHex Encoded Files. P. Faltstrom, D. Crocker, E. Fair. December 1994.
RFC1711	Classifications in E-mail Routing. J. Houttuin. October 1994.
RFC0822/STD0011	Standard for the format of ARPA Internet text messages. D. Crocker. Aug-13-1982.

Mail Transfer

SMTP

Supported Standards

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	Simple Mail Transfer Protocol. J. Postel. Aug-01-1982

Additional Features

SPF Record checking
Relay Restrictions
IP-based Blacklisting
RBL-based blacklisting
Return-Path Verification
Message format correction
Multi-channel delivery
Automated Wake-ups

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

G. Neufeld, J. Baer. July 1998

[RFC1153](#)

Digest Message Format.

F. Wancho. April 1990.

Additional Features

[Index generator](#)

[Subscription Confirmation](#)

[Automated Bounce Processing](#)

Signalling

SIP

Supported Standards

[RFC3903](#)

The Session Initiation Protocol (SIP) Extension for Event State Publication.

A. Niemi, Ed. October 2004.

[RFC3892](#)

The Session Initiation Protocol (SIP) Referred-By Mechanism.

R. Sparks. September 2004.

[RFC3891](#)

The Session Initiation Protocol (SIP) "Replaces" Header.

R. Mahy, B. Biggs, R. Dean. September 2004.

[RFC3863](#)

Presence Information Data Format (PIDF).

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.

J. Rosenberg. August 2004.

[RFC3857](#)

A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP).

J. Rosenberg. August 2004.

[RFC3856](#)

A Presence Event Package for the Session Initiation Protocol (SIP).

J. Rosenberg. August 2004.

[RFC3842](#)

A Message Summary and Message Waiting Indication Event Package for the Session Initiation Protocol (SIP).

R. Mahy. August 2004.

RFC3581	An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing. J. Rosenberg, H. Schulzrinne. August 2003.
RFC3515	The SIP Refer Method. R. Sparks. April 2003.
RFC3428	Session Initiation Protocol (SIP) Extension for Instant Messaging. B. Campbell, Ed., J. Rosenberg, H. Schulzrinne, C. Huitema, D. Gurle. December 2002.
RFC3420	Internet Media Type message/sipfrag. R. Sparks. November 2002.
RFC3372	Session Initiation Protocol for Telephones (SIP-T): Context and Architectures. A. Vemuri, J. Peterson. September 2002.
RFC3327	Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts. D. Willis, B. Hoeneisen. December 2002.
RFC3326	The Reason Header Field for the Session Initiation Protocol (SIP). H. Schulzrinne, D. Oran, G. Camarillo. December 2002.
RFC3325	Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks. C. Jennings, J. Peterson, M. Watson. November 2002.
RFC3311	The Session Initiation Protocol (SIP) UPDATE Method. J. Rosenberg. September 2002.
RFC3265	Session Initiation Protocol (SIP)-Specific Event Notification. B. Roach. June 2002.
RFC3263	Session Initiation Protocol (SIP): Locating SIP Servers. J. Rosenberg, H. Schulzrinne. June 2002.

RFC3262	Reliability of Provisional Responses in Session Initiation Protocol (SIP). J. Rosenberg, H. Schulzrinne. June 2002.
RFC3261	SIP: Session Initiation Protocol. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler. June 2002.
RFC2976	The SIP INFO Method. S. Donovan. October 2000.

Additional Features

[Microsoft SIP Message signing](#)
[application/xpdf+xml](#)
[format](#)

XMPP

Supported Standards

RFC3920	Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. P. Saint-Andre, Ed. October 2004.
RFC3920	Extensible Messaging and Presence Protocol (XMPP): Core P. Saint-Andre, Ed. October 2004.

Simple Notification

Supported Standards

RFC4146	Simple New Mail Notification. R. Gellens. August 2005.
-------------------------	---

Data Access

IMAP

Supported Standards	
<u>RFC3691</u>	Internet Message Access Protocol (IMAP) UNSELECT command. A. Melnikov, February 2004
<u>RFC3516</u>	IMAP4 Binary Content Extension. L. Nerenberg, April 2003
<u>RFC3503</u>	MDN profile for IMAP. A. Melnikov, March 2003
<u>RFC3502</u>	The Internet Message Action Protocol (IMAP4) MULTIAPPEND Extension. M. Crispin, March 2003
<u>RFC3348</u>	The Internet Message Action Protocol (IMAP4) Child Mailbox Extension. M. Gahrns, R. Cheng, July 2002
<u>RFC2971</u>	IMAP4 ID extension. T. Showalter, October 2000
<u>RFC2683</u>	IMAP4 Implementation Recommendations. B. Leiba, September 1999
<u>RFC2595</u>	Using TLS with IMAP, POP3 and ACAP. C. Newman. June 1999.
<u>RFC2359</u>	IMAP4 UIDPLUS extension. J. Myers, June 1998
<u>RFC2342</u>	IMAP4 Namespace. M. Gahrns, C. Newman, May 1998
<u>RFC2221</u>	IMAP4 Login Referrals. M. Gahrns, October 1997

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

Additional Features

Controlled access to external (shared) mailboxes.

Public shared mailboxes.

Support for various Mailbox formats.

POP

Supported Standards	
RFC2595	Using TLS with IMAP, POP3 and ACAP. C. Newman. June 1999.
RFC2449	POP3 Extension Mechanism R. Gellens, C. Newman, L. Lundblade. November 1998.
RFC2384	POP URL Scheme R. Gellens. August 1998.
RFC1939/RFC1725/STD0053	Post Office Protocol - Version 3. J. Myers & M. Rose. May 1996.
RFC1734	POP3 AUTHentication command. J. Myers. December 1994.
Additional Features	
XTND.XMIT	Submitting messages via the POP protocol
LAST command	
Multi-Mailboxes	Accessing all Account Mailboxes using any POP mailer
Multi-Mailboxes	Accessing Public and Shared Mailboxes using any POP mailer
Support for various Mailbox formats.	
RPOP Module	
Polling	Automated Mail Retrieving from Remote Accounts
Unified Domain-Wide Accounts	Automated Mail Distribution of Retrieved Mail

FTP

Supported Standards

RFC2389	Feature negotiation mechanism for the File Transfer Protocol P. Hethmon, R. Elz. August 1998.
RFC2228	FTP Security Extensions M. Horowitz, S. Lunt. October 1997.
RFC0959	FILE TRANSFER PROTOCOL (FTP) J. Postel, J. Reynolds. October 1985.

TFTP

Supported Standards

RFC1350	THE TFTP PROTOCOL (REVISION 2) K. Sollins. July 1992.
-------------------------	--

ACAP

Supported Standards

RFC2595	Using TLS with IMAP, POP3 and ACAP. C. Newman. June 1999.
RFC2244	Application Configuration Access Protocol. C. Newman, J. Myers. November 1997.

WebUser Interface

Supported Standards	
RFC2646	The Text/Plain Format Parameter. R. Gellens. August 1999.
RFC2557	MIME Encapsulation of Aggregate Documents, such as HTML (MHTML). J. Palme, A. Hopmann, N. Shelness. March 1999.
RFC2368	The mailto URL scheme. P. Hoffman, L. Masinter, J. Zawinski. July 1998.
RFC2298	An Extensible Message Format for Message Disposition Notifications. R. Fajman. March 1998.
RFC1896	The text/enriched MIME Content-type. P. Resnick,A. Walker. February 1996.
RFC1844	Multimedia E-mail (MIME) User Agent checklist. E. Huizer. August 1995.
RFC1808	Relative Uniform Resource Locators. R. Fielding. June 1995.
RFC1738	Uniform Resource Locators (URL). T. Berners-Lee, L. Masinter, M. McCahill. December 1994.

MultiMedia

SDP

Supported Standards	
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).

J. Rosenberg, H. Schulzrinne. June 2002.

[RFC2327](#) SDP: Session Description Protocol.

M. Handley, V. Jacobson. April 1998.

RTP

Supported Standards

[RFC3555](#) MIME Type Registration of RTP Payload Formats.

S. Casner, P. Hoschka. July 2003.

[RFC3551](#) RTP Profile for Audio and Video Conferences with Minimal Control.

H. Schulzrinne, S. Casner. July 2003.

[RFC3550](#) RTP: A Transport Protocol for Real-Time Applications.

H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. July 2003.

[RFC2833](#) RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals.

H. Schulzrinne, S. Petrack. May 2000.

Voice and Video Mail

Supported Standards

[RFC4024](#) Voice Messaging Client Behaviour.

G. Parsons, J. Maruszak. July 2005.

[RFC3938](#) Video-Message Message-Context.

T. Hansen. October 2004.

[RFC3458](#) Message Context for Internet Mail.

E. Burger, E. Candell, C. Eliot, G. Klyne. January 2003.

[RFC2423](#) VPIM Voice Message MIME Sub-type Registration.

G. Vaudreuil, G. Parsons. September 1998.

[RFC2421](#) Voice Profile for Internet Mail - version 2.
G. Vaudreuil, G. Parsons. September 1998.

Groupware

Calendar and Tasks

Supported Standards

[RFC3283](#) Guide to Internet Calendaring.
B. Mahoney, G. Babics, A. Taler. June 2002.

[RFC2447](#) iCalendar Message-Based Interoperability Protocol (iMIP).
F. Dawson, S. Mansour, S. Silverberg. November 1998.

[RFC2446](#) iCalendar Transport-Independent Interoperability Protocol (iTIP).
S. Silverberg, S. Mansour, F. Dawson, R. Hopson. November 1998.

[RFC2445](#) Internet Calendaring and Scheduling Core Object Specification (iCalendar).
F. Dawson, D. Stenerson. November 1998.

Contacts

Supported Standards

[RFC2426](#) vCard MIME Directory Profile.
F. Dawson, T. Howes. September 1998.

[RFC2425](#) A MIME Content-Type for Directory Information.
T. Howes, M. Smith, F. Dawson. September 1998.

Additional Features

vCard 2.1

Services

HTTP

Supported Standards	
RFC2818	HTTP Over TLS E. Rescorla. May 2000.
RFC2617	HTTP Authentication: Basic and Digest Access Authentication J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart. June 1999.
RFC2616	Hypertext Transfer Protocol -- HTTP/1.1. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee. January 1997.
RFC2388	Returning Values from Forms: multipart/form-data L. Masinter. August 1998.
RFC2109	HTTP State Management Mechanism D. Kristol, L. Montulli. February 1997.
Additional Features	
Page Caching	Internal Cache for Web Pages and Forms

LDAP

Supported Standards	
RFC2891	LDAP Control Extension for Server Side Sorting of Search Results T. Howes, M. Wahl, A. Anantha. August 2000.
RFC2849	The LDAP Data Interchange Format (LDIF) - Technical Specification G. Good. June 2000.

RFC2830	LDAPv3: Extension for Transport Layer Security J. Hodges, D. Byrne, B. Blakley, P. Behera. May 2000.
RFC2829	Authentication Methods for LDAP M. Wahl, H. Alvestrand, J. Hodges, R. Morgan. May 2000.
RFC2820	Access Control Requirements for LDAP E. Stokes, R. Morgan, M. Wahl. May 2000.
RFC2798	Definition of the inetOrgPerson LDAP Object Class. M. Smith. April 2000.
RFC2696	LDAP Control Extension for Simple Paged Results Manipulation A. Herron, A. Anantha, T. Howes. September 1999.
RFC2587	Internet X.509 Public Key Infrastructure. LDAPv2 Schema S. Boeyen, T. Howes, P. Richard. June 1999.
RFC2256	A Summary of the X.500(96) User Schema for use with LDAPv3. M. Wahl. December 1997.
RFC2255	The LDAP URL Format. T.Howes, M.Smith. December 1997.
RFC2254	The String Representation of LDAP Search Filters. 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	
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.
RFC1212	Concise MIB Definitions. Rose, M., and K. McCloghrie. March 1991

RADIUS

Supported Standards

RFCdraft	RADIUS Extension for Digest Authentication B. Stermann, D. Sadolevsky, D. Schwartz, D. Williams, W. Beck. February, 2005.
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) G. Zorn. March 2001.
RFC2869	RADIUS Extensions C. Rigney, W. Willats, P. Calhoun. June 2000.
RFC2868	RADIUS Attributes for Tunnel Protocol Support G. Zorn, D. Leifer, A. Rubens, J. Shriver, M. Holdrege, I. Goyret. June 2000.
RFC2866	RADIUS Accounting C. Rigney. June 2000.
RFC2865	Remote Authentication Dial In User Service (RADIUS) C. Rigney, S. Willens, A. Rubens, W. Simpson. June 2000.
RFC2759	Microsoft PPP CHAP Extensions, Version 2 G. Zorn. January 2000.
RFC2548	Microsoft Vendor-specific RADIUS Attributes. G. Zorn. March 1999.
RFC1994	PPP Challenge Handshake Authentication Protocol (CHAP) W. Simpson. August 1996.

BSD syslog

Supported Standards

[RFC3164](#) The BSD syslog Protocol.
C. Lonvick. August 2001.

DNR

Supported Standards

[RFC3761](#) The E.164 to Uniform Resource Identifiers (URI)
.Dynamic Delegation Discovery System (DDDS) Application (ENUM)
P. Faltstrom, M. Mealling. April 2004.

[RFC3596](#) DNS Extensions to Support IP Version 6
S. Thomson, C. Huitema, V. Ksinant, M. Souissi. October 2003.

[RFC2916](#) E.164 number and DNS.
P. Faltstrom. September 2000.

[RFC2915](#) The Naming Authority Pointer (NAPTR) DNS Resource Record.
M. Mealling, R. Daniel. September 2000.

[RFC2782](#) A DNS RR for specifying the location of services (DNS SRV).
A. Gulbrandsen, P. Vixie, L. Esibov. February 2000.

[RFC1035](#) Domain names - implementation and specification.
P.V.Mockapetris. Nov-01-1987.



Revision History

Current Versions Release History

5.1c1 05-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: Directry 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 Refreshers 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.

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: misformed 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 cause 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 used 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.

The 5.0 Branch has been switched to *Stable*.

Note: mailboxes with voicemail messages (messages with the Media flag set) are not readable with pre-5.0 versions. Remove all voicemail messages from your mailboxes if you need to downgrade.

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 "IG" 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.

5.0.2 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.

5.0.1 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 "gate-

way 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 signalling 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 is 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.



How To

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 Names is:	<div>Rerouted to   <input type="text" value="*%domain.dom@otherserver.dc"/></div>

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/mailling 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 server `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.

SMTP Delivery

How can I relay mail for certain domains?

If you want your Server to act as a back-up mail relay for certain domains, you can enable the `Relay to All Hosts We Backup` option in the [SMTP module](#) settings. But this is not a perfect solution, since anybody with access to any DNS server would be able to use your server for unauthorized relaying.

To safely back-up 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 mail for the `friend.com` domain, but it should go to via a different server firewall.friend.com, use the following Router record:

```
Relay: friend.com = friend.com@firewall.friend.com.via
```

If you want to bypass the MX records and relay all mail to a certain IP address (specified explicitly or

using a DNS A-record), then see the [Bypassing MX](#) section.

How can I send mail to a remote host bypassing its DNS MX records?

If your server should send mail to a domain `target.domain` via the relay `relay.domain`, you can specify the IP address of that relay in the [Router](#):

```
target.domain = target.domain@[11.22.33.44]
```

You may want to relay mail using DNS A-records instead of explicitly specified IP addresses:

```
target.domain = target.domain@relay.domain.25.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.

Note: You may want to add a `Relay:`, `NoRelay:` or `RelayAll:` prefix

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/PROPOP)

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, you 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?

In a corporate environment, it may be necessary to store all outgoing mail into a mailbox in a system administrator or a security officer account.

Note: if your company chooses to copy employee mail, it **MUST** notify all server users about this policy.

To copy mail sent from certain domains, use a Server-wide [Rule](#):

Data	Operation	Parameter
Return-Path	is	*@mydomain.dom
Action	Parameters	
Store in	~security/outgoing	

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?

In a corporate environment, it may be necessary to let certain groups of users send mail only to other members of that group and to only certain addresses outside that group.

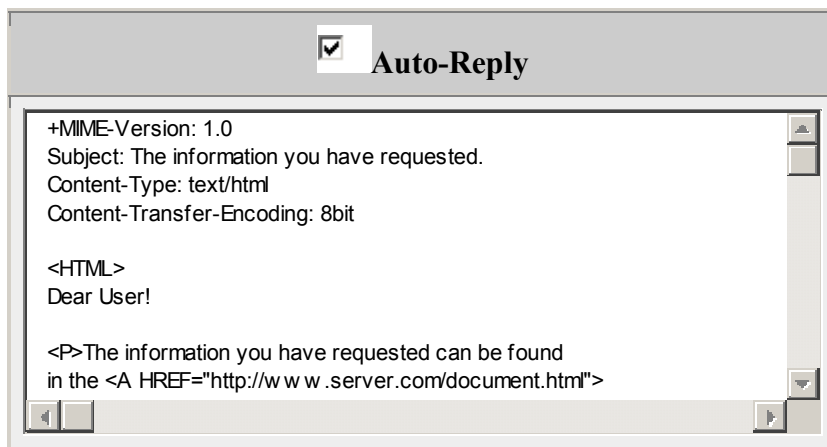
The simplest way to implement restrictions is to organize these groups of users into CommuniGate Pro Domains. If all users in the Domain dept1.company.dom (except the user boss) are allowed to send mail only to the users in the same Domain and to the supervisor@hq.company.dom address, then the following Server-wide Rule should be used:

Data	Operation	Parameter
Return-Path	is	*@dept1.company.dom
Return-Path	is not	boss@dept1.company.dom
Any Recipient	not in	*@dept1.company.dom,supervisor@hq.co

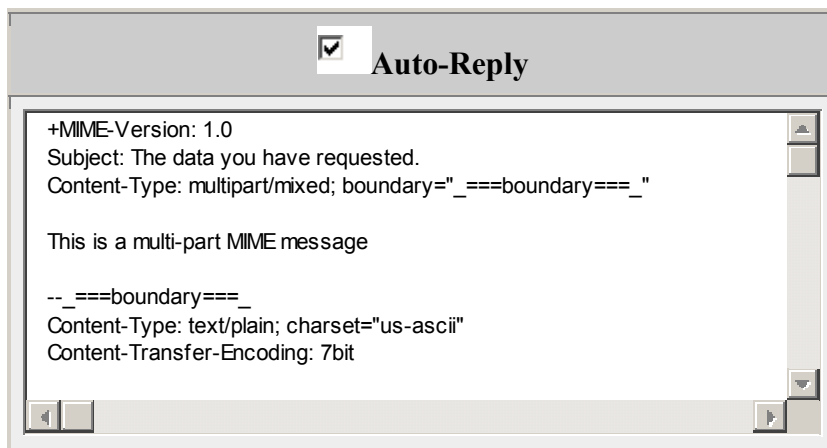
Action	Parameters
Reject with	you can send mail only to dept1.company.

How can I create an autoresponder that sends files or HTML messages?

You can use Rule "Reply" actions or the simplified AutoResponder Rule to generate messages in any MIME format. Just start the Reply text with the plus (+) sign and add all necessary MIME headers. Rememeber that the Subject field is not autogenerated in this case and that you have to specify the MIME-Version: header field, too.



You can use the same method to send non-text attachments:



The easiest way to compose such a message is to send the required file to your CommuniGate Pro account using MIME-encoding, and then open the message using the WebUser Interface. After verifying that the message has arrived intact, click the "Undecoded Letter" icon in the message header panel.

The undecoded text of the message will be displayed in a new browser window. You can copy the encoded message body text and paste it to the Rule text field.

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

accountmarketing, 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 accountmarketing, 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 Accounts [access right](#) has unlimited access rights to all mailboxes in all accounts on the Server. 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 client to access user mailboxes. That client should be able to let users enter the mailbox name directly. To open the INBOX in the username account, administrators should log in under their own names and tell the IMAP 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.

How can I provide username.domain.dom personal File Sites?

The standard URL for a Personal File Site of the username@domain.dom account is `http://domain.dom/~username`.

You may want to provide better looking `http://username.domain.dom/` URLs for your Account Personal File Sites. This feature is based on the method the CommuniGate Pro Server uses to process [HTTP requests](#) sent to the WebUser port(s).

For users in a secondary domain domain.dom, 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 have to 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.



Help Me

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.

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.

You have selected the Reject all Logins from Non-Client Addresses option on the Protection page. Now you 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.

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 defaults to the main domain 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 IP Addresses setting set to "All Available", so all addresses not assigned to secondary domains are automatically assigned to the Main Domain. If none of your Domains has the 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 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 be found in the Directory Account Properties, 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:

```
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.

Misc

What is that 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

You should download the CommuniGate Pro software either from the Stalker [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 queued messages, account files, settings, and other server data.

Installing on a Sun Solaris System.

- 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` 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 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 sendmail), stop that server (for example, you can use the `/sbin/chkconfig sendmail off` command and then restart the server computer).
- 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 sign (#) 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 an early, 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.

If you are running a modern version of Linux, and your OS manufacturer has ensured you that the NPTL library they ship is stable, you may want to remove this command from the startup script.

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.

Note: Linux kernels do not support x86 hyperthreading correctly. Make sure hyperthreading is switched off in the x86 server BIOS.

Installing on a MS Windows 200x/XP/NT/9x/ME 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 "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/2000/XP

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 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 10.2 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/sbin/` directory.
- Note: The Installer creates the startup directory `/System/Library/StartupItems/CommuniGatePro`, so the CommuniGate Pro Server will auto-start when the MacOS X System starts.
- Note: The Installer packs the startup directory `/System/Library/StartupItems/Sendmail` into the `/System/Library/StartupItems/Sendmail.tar` startup archive, so the legacy `sendmail` daemon will not auto-start. If you decide to uninstall CommuniGate Pro, this archive will be unpacked and the Sendmail startup directory 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 `/System/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 4.x (supporting FreeBSD 4.0 and up), one - for FreeBSD 5.x (supporting FreeBSD 5.3 and better).

- Log in as a super-user (root).
- Install the CommuniGate Pro package. FreeBSD 4.x:
`pkg_add CGatePro-FreeBSD4-version.tgz`

FreeBSD 5.x:

```
pkg_add CGatePro-FreeBSD5-version.tgz
```

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 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 `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 `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-version.tar.gz
tar -xpf CGatePro-BSDI-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-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 `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 the HP/UX version 11 is installed.
- Apply all patches available for HP-UX 11.00 or at least the threads-related ones.
- 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 -xzipf CGatePro-HPUX-HPPA-version.tar.gz
tar -xf CGatePro-HPUX-HPPA-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 "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-Tru64-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-version.tar.gz
tar -xpf CGatePro-UnixWare-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-version.tar.gz  
tar -xpf CGatePro-OpenServer-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 V4R4M0 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 CommuniGatePro 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:

```
CHGJOB JOB (CGSERVER/CGSERVERJD) RQSDTA ('QGPL/STRCGSRV')
```

Add an autostart job entry:

```
ADDAJE SBS (QSYSWRK) JOB (CGSERVER) JOB (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-Vversion-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:SYSHUTDOWN.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 startup 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) `guntar` command:

```
guntar 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

- Restart the system or start the CommuniGate Pro Server manually.
- On a Unix or QNX system: use either the super-user account or any account included into the `mail` group to access the CommuniGate Pro files in the *base directory* (by default that directory is `/Local/CommuniGate` for MacOS X Rhapsody, `/var/CommuniGate/` for other Unix systems).
On a OpenVMS system: use the `SYSTEM` account to access the CommuniGate Pro files in the *base directory* (by default, that directory is `SYSS$SPECIFIC:[CommuniGate]`).
On a Unix, QNX, or BeOS system, open the file:
`{CGateBase}/Accounts/postmaster.macnt/account.settings`
- On a MacOS X (Darwin) system:
find the `ShowPassword.command` file unpacked alongside the `CommuniGate.pkg`

package file and double-click it.

or

open the Terminal application and type the following command:

```
sudo more /var/CommuniGate/Accounts/postmaster.macnt/  
account.settings
```

in both cases you will be asked for the MacOS X "root" password.

On a MS Windows or an OS/2 system, open the file:

```
{CGateBase}\Accounts\postmaster.macnt\account.settings
```

On an OS/400 system, use any ASCII-capable text editor to open the file:

```
{CGateBase}/Accounts/postmaster.macnt/account.settings
```

For example, you can use the following command:

```
EDTF STMF('/var/CommuniGate/Accounts/postmaster.macnt/  
account.settings')
```

On an OpenVMS system, open the file:

```
SYS$SPECIFIC:[CommuniGate.Accounts.postmaster-  
macnt]account.settings
```

For example, you can use the following command:

```
type SYS$SPECIFIC:[CommuniGate.Accounts.postmaster-  
macnt]account.settings
```

This text file contains a random password assigned to the `postmaster` account. Remember this password, and do not change it at this time (you will be able to change the `postmaster` account password via the WebAdmin Interface).

- Connect to the server 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.

- To open Setup, Accounts, Monitor, and Master configuration pages (realms), use the `postmaster` username and the password you have seen in the `postmaster.account.settings` file.
- Check the [General Settings](#). Make sure the date/time settings of the Server operating system are set properly.
- Proceed using the [Quick Start](#) instructions.

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, Personal File Sites, Licenses, customized WebUser and WebAdmin files stay in place and continue to work with the new version of the CommuniGate Pro software.

Note: if you chose to manually modify WebUser and/or WebAdmin 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 `WebUser` and `WebAdmin` directories inside the `Account` and `Domains` subdirectories - graphic files used in the customized `WebUser` and `WebAdmin` Interfaces. Binary files can also be stored in `Personal File Sites` - `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 recoding).

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.



Quick Start

This section outlines the basic CommuniGate Pro configuration options and provides references to in-depth information.

After the CommuniGate Pro Server software is [Installed](#), take the following steps to get a working communication system.

When the CommuniGate Pro Server is running, and you should be able to access its WebAdmin Interface by pointing your browser to http://your_sever_address:8010, and you should already know the `postmaster` password.

If you have any problem with configuration, check the [How To](#) and [Help Me](#) sections.

Postmaster Password

Open the Accounts WebAdmin Realm and click the `postmaster` link to open the postmaster Account Settings. Specify a new password and click the Update button. The next attempt to access any WebAdmin page will result in a delay, and the browser will ask you to enter the new password.

Main Domain Name

Open the Settings WebAdmin Realm and open the General section. Specify the proper name in the Main Domain Name field and click the Update button. [[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 may have its own Accounts, Settings, etc. [[read more](#)]

Network

Open the Settings WebAdmin Realm and open the Network section. If your Server is connected to your office LAN, specify that LAN network addresses. [[read more](#)]

Locale

Open the Domains WebAdmin Realm and open the Account Defaults section and switch to the WebUser Preferences section. 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. Select the Default Time Zone. [[read more](#)]

Accounts

Open the Accounts WebAdmin Realm and create (*register*) your Server users. [[read more](#)]



Migrating to CommuniGate Pro

If your system was already running some type of 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 [External INBOX](#) option. In this case, the account INBOX will not be created inside the CommuniGate Pro *base directory*. Instead, the mailbox location will be taken from the [user 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 *external 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 [Sharing](#) 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 external 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 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 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-crypt` (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](#) interface, and the Unix (crypt-encrypted) password for that user is `As1UzT1JkPsocc`, then use the following CLI command:

```
createaccount "test" {Password="\002AslUzTlJkPsocc";}
```

If you create users by [importing](#) an account list from a text file, place the Unix passwords strings into the `Unix-Password` 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](#) 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](#).

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).
- `{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.

Sample Import file:

Name	UnixPassword	Password Type
user2	YIdhkjeHDKbYsji	Unix-crypt
user1	{SHA}Ue4Erbim2TC7CmuukMOBejeytr2=	SHA1-digested
user3	{MD5}zverMUhsgJUIDjeytr2=	MD5-digested
user4	{crypt}YIdhkjeHDKbYsji	Unix-crypt, same as for the user1 account
user5	\$1\$VlPrB\$vNjOAytB3W.j0bkbaN2Z.	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@domain.dom` address to `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).

Restricted Relaying

Unlike `sendmail`, CommuniGate Pro does not use syntax rules to prevent [unauthorized relaying](#).

Instead, it uses the [Router](#) and really checks if message delivery to a specified address would result in SMTP transfer to a "stranger" host.

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 POPpass-  
word 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 sign: `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 sign: `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 new-  
Password
```

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 sign: `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 sign: 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.

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.

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 [Account Import](#) to a CommuniGate 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 *SMTPPrecipient* 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.

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 encrypted password your found in that account record is `/etc/shadow` 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 Domain, and in the Domain settings enable the `Consult External Authenticator` option. 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 CommuniGatePro 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 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.



System Administration

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).

Note: If you use a Netscape® browser, check that its caching setting (Preferences->Advanced->Cache) is set to Every time.

Sections and Privileges

The WebAdmin Server administration pages are divided into four groups (realms). To access a page in any group, a user should be registered with the CommuniGate Pro Server (should have an Account on the Server), and the user should be explicitly granted access rights to that section.

- The Settings section contains pages that allow a Server Administrator to modify the Server kernel and module settings.
- The Accounts and Domains section contains pages that allow a Server Administrator to create and remove secondary domains and accounts, and to modify the domain and account settings.

- The Directory section contains pages that allow a Server Administrator to configure the Directory services of the CommuniGate Pro server.
- The Monitors section 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 Server [statistics](#).

If a user is granted an access right to the Monitors section, additional Monitor Access rights can be granted, too (rights to release and reject module queues, reconfigure the Log Manager, etc.)

- The Master section contains the pages that allow a Server Administrator to grant and revoke Server Administrator access rights, and to modify the Server License Keys.

Note: If a user is granted the Master access right, that user can access all other sections.

Note: These access rights can be granted to the Accounts (users) in the Main Domain 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, assigns a random password to that Account, and grants the Master access right to the `postmaster` user.

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 [Secondary 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.

- **Directory** file. This file contains or describes the Server [Central Directory](#).

For more information about the Account and Domain files and directories, see the [Account Data](#) section.

You can use symbolic links to move some of these directories to other locations (and other disks).

General Settings

Start configuring the Server by opening the General page in the Settings section.

Main Domain Name:	<input type="text" value="mycompany.com"/>
System Internals Log:	<input type="text" value="Major & Failures"/> ▼
Crash Recovery:	<input type="text" value="Enabled"/> ▼
Server Time:	Wed, 21 Jun 2006 19:55:22 -0700
Server Up-Time:	118 days, 0h 1m 3s
Server OS:	Sun Solaris <input type="button" value="Drop Root"/>
Server Hardware:	x86
Server Version:	5.1
MAPI Version:	1.1.44
IPv6 Support:	Enabled
Server IP Address(es):	[216.200.213.118],[10.0.0.5],[2001:470:1f01:2565::a:845] <input type="button" value="Refresh"/>
Name Server(s) IP Address(es):	[216.200.213.113],[216.200.213.114]

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 name looks like `host12345hh.com-company.com`, you should change it to the name of the domain this Server should process.

Note: unless you create additional [Domain Administrator access right](#) ONLY the messages 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` will not be processed as local, and if such a message is received, the server will try to deliver it to the `mail.company.com` system over the network. If the DNS record for the `mail.company.com` points to the same Server computer, the *mail loop* error will be detected, and the message will be rejected.

If your server should process mail for several domains, 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:

A server should process mail for the `company.com` and `client1.com` domains. In the DNS system, these domain names have only MX-records pointing to `mail.company.com` and `mail.client1.com` A-records, and these A-records point to IP address(es) belonging to the 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 Aliases fields.
- enter `mail.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 Aliases fields.
- enter `mail.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 Stalker ftp site for examination.

Stalker Software recommends you to disable this option if you are running any beta-version of the CommuniGate Pro software.

Information fields

Information fields on the General Settings page display the name of the Server Operating System, the hardware platform, the version of the CommuniGate Pro Server, the version of the [MAPI Connector](#) server part, the Server network address(es), 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.

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 Server 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.

Open the Account Defaults page in the Domains section of the WebAdmin Interface if you want to set the Server-wide default language. If you want to set a default language 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 WebUser Preferences link to open the Default WebUser 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 WebUser Preferences link to open the Default WebUser Preferences page.

Select the default Time Zone from the list. If you select the "built-in" zone (**), 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.

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.

`--noLockFile`

This option tells the Server not to create 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.

`--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 NT/2000/XP 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. This option is enabled by default on the Microsoft Windows 9x/ME and IBM OS/2 platform.

`--useNonBlockingSockets`

This option tells the Server to set its TCP/IP 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/IP 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).

`--NoThreadPriority`

This option tells the Server to skip all attempts to increase individual thread priority. Use this option if bugs in 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.

`--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).

`--IPv6 [YES | NO]`

See the [Network](#) section for the details.

Command line option names are case-insensitive.

Specifying Command Line Options under Windows NT/2000/XP

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 "-" sign 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

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.

- 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 Administration

If your Server has several [Domain Administrator access right](#), 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 Admininstrator 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 (*account-Name@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	WebUser Interface: Auto-Signup Setting
TrailerText	WebUser Interface: 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 Authenticator 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
CentralDirectory	Directory Integration Setting
CertificateType	Security: Domain PKI Settings
KerberosKeys	Security: Kerberos Keys
RelayAddress	SMTP Sending: Send via Setting
recipientStatus	SMTP Receiving: When Receiving Setting

Objects

Access Right	Description
CanCreateAccounts	Create , rename , and remove Accounts
CanCreateGroups	Create, rename, remove, and modify Groups
CanCreateForwarders	Manage Forwarders
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 Personal File Sites
CanCreateWebUserSessions	Manage WebUser sessions via CLI
CanImpersonate	Ability to Impersonate

Account Settings


Access Right	Description
BasicSettings	Basic Settings: Password, RealName, Custom and Public Info settings
WebUserSettings	WebUser Interface Settings
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
AccessModes	Enabled Services
MaxAccountSize	Mail Storage limits: Mail Storage
MaxMailboxes	Mail Storage limits: Mailboxes
DefaultMailboxType	Mail Storage options: New Mailboxes
QuotaNotice	Mail Quota Processing: Send Notice
QuotaAlert	Mail Quota Processing: Send Alerts
QuotaSuspend	Mail Quota Processing: Delay New Mail
RulesAllowed	Mail Processing: Rule
RPOPAllowed	Mail Processing: RPOP Accounts
MailToAll	Mail Processing options: Accept Mail to all
AddMailTrailer	Mail Processing options: Add Mail Trailer
SignalRulesAllowed	Signal processing options: Rules
MaxSignalContacts	Signal processing limits: Contacts
MaxWebSize	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 Accounts to be displayed in the Account 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 cor-

rect character set first.

Each CommuniGate Pro WebAdmin *realm* has its own WebAdmin Preferences page. Click the  icon on any of the WebAdmin pages 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. By uploading custom `admin*` files to the default Domain Skin, the Domain WebAdmin Interface is modified.

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 Interface. The Server Administrator will see the framed version of the 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 mailers.

To modify the Server Strings, the administrator should follow the [Strings](#) link on the [General](#) Settings page. The Server Strings page appears (the actual page has much more strings):

Keyword	String	
POPPrompt	<input type="radio"/>	<input type="text"/>
	<input checked="" type="radio"/>	CommuniGate Pro POP3 Server ^0 ready
SMTPByeBye	<input type="radio"/>	<input type="text" value="is shutting up"/>
	<input checked="" type="radio"/>	CommuniGate Pro SMTP closing connection
SMTPNoRelay	<input type="radio"/>	<input type="text"/>
	<input checked="" type="radio"/>	we do not relay
SMTPNonInternet	<input type="radio"/>	<input type="text"/>
	<input checked="" type="radio"/>	will leave the Internet
SMTPNormalPrompt	<input type="radio"/>	<input type="text"/>
	<input checked="" type="radio"/>	^1 ESMTP CommuniGate Pro ^0







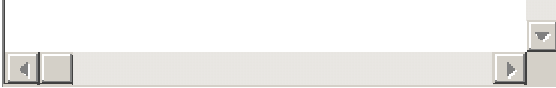
To modify a Server String, enter the new text in the text field, and select the upper radio button. To change the string to its default value (displayed under the text field), simply select the lower radio button.

Click the Update button to update the Server Strings.

Domain Name Resolver (DNR)

The CommuniGate Pro server uses its own high-speed multithreaded Domain Name Resolver to convert domain names into network (IP) addresses. To convert names, the Domain Name Resolver sends requests to the specified Domain Name Servers.

Server Administrators with the Can Modify Settings access right can modify the Resolver settings. Open the Obscure page in the Settings section of the Server WebAdmin Interface:

Domain Name Resolver	
Log:	Problems  Concurrent Requests: 10 
Initial Time-out:	7 seconds  Retry Limit: 5 
DNS Addresses:	OS-specified  [209.1.58.247], [206.40.74.1]
Dummy IP Addresses:	64.94.110.11 ; *.com, *.net "wildcard"  

Log

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 relay processing several hundred requests per second, this parameter should be selected after some testing: older DNS servers may crash if requested to process too many concurrent requests, also in certain cases the DNR traffic may start to compete with the mail transfer (SMTP) traffic.

Initial Time-out

By default Domain Name System requests are sent via UDP, so request or response packets can be lost. This option specifies the time period the Domain Name Resolver will wait for a response from a DNS server.

If a response is not received, the Resolver resends the request, and waits twice the initial time period, if it times out again, it can resend the request again and it will wait three times longer than the initial time period.

If you have several Domain Name Servers specified, each time the Resolver needs to resend a request, it sends it to the next DNS server in the list.

Retry Limit

This option specifies how many times 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 Resolvers sends the same request not more than twice, and both times it uses the same (Initial) response time-out.

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 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.

Dummy IP Addresses

This setting allows you to specify network (IP) addresses and/or address ranges 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 uses TCP connections if the server UDP response came back with the "Truncated" flag set. This feature allows the Resolver to retrieve very large records from DNS servers.

External Helper Programs

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 should support the simple Helper Interface.

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:

<input checked="" type="checkbox"/> <i>Helper Name</i>	
Log:	Major & Failures ▼
Program Path:	/usr/local/sbin/pasw check
Time-out:	disabled ▼
Auto-Restart:	15 seconds ▼

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*.

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.



Server Logs

All components of the 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 Stalker Technical Support, always include a portion of the Log that indicates the problem.

Creating and Deleting Log Files

You can use any Web browser to examine the Logs. Click the Logs button in the Monitor section and the list of the stored Logs appears. The current Log is marked with the asterisk (*) sign.

You should have the "Can Monitor" [Server Access Right](#) to view the Logs.

The options on the top of the page allow you to specify when the Logs files are created and deleted:

Log Manager Options			
Start New File	Every:	<input type="text" value="day"/>	or if Larger than: <input type="text" value="1000K"/>
Active Log	Open showing last:	<input type="text" value="2 minutes"/>	Delete Old Files in: <input type="text" value="10 days"/>
External Logger	Records to send:	<input type="text" value="All Info"/>	Server address: <input type="text" value="[10.0.34.56]"/>
		<input type="button" value="Reset"/>	<input type="button" value="Update"/>

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.

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 "CanTuneLoggerSettings" [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:

Display	Filter: <input type="text"/>	11 selected
	Name	Size
*	2001-04-04	115K
<input type="checkbox"/>	2001-04-03	204K
<input type="checkbox"/>	2001-04-02	34K
<input type="checkbox"/>	2001-04-01	34K
<input type="checkbox"/>	2001-03-31	34K
<input type="checkbox"/>	2001-03-30	25K
<input type="checkbox"/>	2001-03-29	33K
<input type="checkbox"/>	2001-03-28	35K
<input type="checkbox"/>	2001-03-27	34K
<input type="checkbox"/>	2001-03-26	33K
<input type="checkbox"/>	2001-03-25	33K
Delete Marked Logs		

Open showing last

When you open the currently active Log file, this setting specifies the initial 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.

Click the Log file name to open the selected Log.

Specifying a Time Interval

When the Log appears in your browser window, all Log records are displayed. Since Logs can have thousands of

them, you may want to view only a portion of the Log. Interrupt the Log downloading process and specify the Log Level and the Time Range options:

The screenshot shows a dialog box titled "Selection". It contains a "Display" button on the left. To its right are three rows of controls:

- Level:** A dropdown menu currently showing "All Info".
- Filter:** A text input field followed by two checkboxes labeled "Keyed" and "RegEx".
- Interval:** Two time input fields, the first containing "10:20:00" and the second containing "10:30:00", separated by a minus sign.

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.

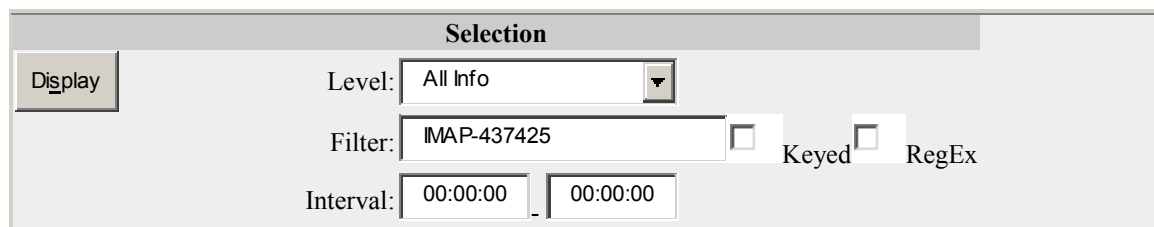
RegEx

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.



Selection

Display

Level: All Info

Filter: IMAP-437425 ☒ Keyed ☐ Regex

Interval: 00:00:00 - 00:00:00

```
00:06:23.261 4 IMAP-437425([64.173.55.175]) got connection on
[64.173.55.169:143] (mail.stalker.com) fr
00:06:23.261 5 IMAP-437425([64.173.55.175]) out: * OK CommuniGate Pro IMAP Server 5.1.8 at
mail.stalke
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 AAAAAAAAAAAAAAAAAAAAAA=
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:VCALEN-
DAR\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:

Selection			
Display	Level:	All Info	
	Filter:	: Call-ID: 72D532E1CEB813B537	<input checked="" type="checkbox"/> Keyed <input type="checkbox"/> RegEx
	Interval:	00:50:00	

```
00:54:10.312 2 SIPDATA-000502 out: req udp [10.0.0.1]:5060 REGISTER(680 bytes)
sip:node6.stalker.com
00:54:10.312 5 SIPDATA-000502 out: REGISTER sip:node6.stalker.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:username@node6.stalker.com>
00:54:10.312 5 SIPDATA-000502 out: Call-ID: 72D532E1CEB813B537E4E44058354C68-
2494453@node9.stalker.com
00:54:10.312 5 SIPDATA-000502 out: Contact: <sip:299@node9.stalker.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.stalker.com",username="usr-
name",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:username@node6.stalker.com>;tag=9B5A8DB531C3FD7A
00:54:10.328 5 SIPDATA-000503 inp: To: <sip:username@node6.stalker.com>;tag=7FBB267A3903E5B0
00:54:10.328 5 SIPDATA-000503 inp: Call-ID: 72D532E1CEB813B537E4E44058354C68-
2494453@node9.stalker.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.stalker.com;services=no>;expires=90
```

```
00:54:10.328 5 SIPDATA-000503 inp: Event: registration
00:54:10.328 5 SIPDATA-000503 inp: Date: Thu, 16 Mar 2006 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.stalker.com
00:55:25.328 5 SIPDATA-000507 out: REGISTER sip:node6.stalker.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:username@node6.stalker.com>;tag=35270A39FB68F573
00:55:25.328 5 SIPDATA-000507 out: To: <sip:usrname@node6.stalker.com>
00:55:25.328 5 SIPDATA-000507 out: Call-ID: 72D532E1CEB813B537E4E44058354C68-
2494453@node9.stalker.com
00:55:25.328 5 SIPDATA-000507 out: Contact: <sip:299@node9.stalker.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.stalker.com",username="usr-
name",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:username@node6.stalker.com>;tag=35270A39FB68F573
00:55:25.343 5 SIPDATA-000508 inp: To: <sip:username@node6.stalker.com>;tag=7EF99B799DFD7632
00:55:25.343 5 SIPDATA-000508 inp: Call-ID: 72D532E1CEB813B537E4E44058354C68-
2494453@node9.stalker.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.stalker.com;services=no>;expires=90
00:55:25.343 5 SIPDATA-000508 inp: Event: registration
00:55:25.343 5 SIPDATA-000508 inp: Date: Thu, 16 Mar 2006 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 signs (***).

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.

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.

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 switches the Records to Send option to Nothing and it stops sending Log records to the remote syslog server.



Router

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 mail and signals to that Account, and the Alias name can be used to log into that Account.

Address Structure

Each E-mail or Rela-time 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 a 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`.

Mail 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 `stalker.com`, then the following addresses will be converted as shown:

Address	local part	domain part
<code>support@stalker.com</code>	<code>support</code>	<code>stalker.com</code>
<code>-- routed --></code>	<code>support</code>	

<@stalker.com:sales@example.com>	sales@example.com	stalker.com
-- routed -->	sales@gamma.com	
-- routed -->	sales	example.com

Domains and DNS records

Your Server can support many independent [Domians](#) 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 grlobal DNS system.

Example 1: your server (example.com) serves the example.com Domain and a parteners-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 1: 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 messages sent to the client1.com domain will be routed to the Unified Domain-Wide Account client1, etc.

You should also ensure that when a message is sent to the client1.com domain, that message 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.

Open the Router page in the Settings section of the WebAdmin Interface to manage the Routing Table:

```

fax.stalker.com      = stalker.com      ; -> to the same domain
hq.stalker.com       = newhq.stalker.com ; -> to some other server
Relay:*.test.com     = stalker.com      ; aaa.test.com, bbb.test.com
; just a comment line
<sales>              = john              ; simple alias
<sales@client1.com>  = sales-client1     ; simple foreign alias
<info@client1.com>   = info@otherhost.com; account -> other account
<*@client2.com>      = *.cl2             ; sales@.. -> sales.cl2
test.com             = Unified.local     ; unified Domain-Wide Account

```

☐ Add to Non-Qualified Domain Names

Each line in the Routing Table is a routing record. A routing record contains optional prefixes, a left part, the equals sign (=) and a right part. The semicolon sign (;) 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 sign.

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

The CommuniGate Pro Router has the Log Setting that you can modify using a Web browser. Use this setting to specify what kind of information the CommuniGate Pro Router should put in the System Log. Usually you should use the `Major` (address routing) 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 `Relay:` prefix is assumed.

A Routing record can contain one of the following operation-type prefixes:

- `Mail:` (can be shorten to `M:`). Records with this prefix take effect only when an address is routed for some E-mail delivery operation.

- **Access**: (can be shorten to **A:**). Records with this prefix take effect only when an address is routed for some account access operation.
- **Signal**: (can be shorten to **S:**). Records with this prefix take effect only 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.

Sample 1:

`sta*r`

This sample matches any `staXXXXXXr` string where `XXXXXX` is any substring (including an empty substring of 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.

Sample 1:

`sta(3*)r`

This sample matches any `staXXXr` string where `XXX` are any 3 symbols.

Sample 2:

`sta(4+d)r`

This sample matches any `staDDDDDr` string where `DDDDD` are 4 or more decimal digits.

`sta(3-5h)r`

This sample matches any `staHHHr` 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 a Record Sample matches an address, the address is changed to the Record Route. The Substring matching the Sample wildcard substitutes the Route wildcard.

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.stalker.com = twisted.stalker.com
```

All addresses with the `hq.stalker.com` domain part are changed to have the `twisted.stalker.com` domain part. The Router restarts, trying to route the modified address.

A routing path can specify relays.

Example:

```
hq.stalker.com = hq.stalker.com@relay.stalker.com
```

All messages and signal directed to the domain name `hq.stalker.com` will be redirected to the system (domain) `relay.stalker.com`, and then, from that system, to the domain `hq.stalker.com`.

If mail to several domains should be routed in the same or similar way, you may use the asterisk sign 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 messages and signals sent to all subdomains of its Main Domain as messages 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 a message 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 E-mail 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 E-mail 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 messages sent to dept-sales@mycompany.com to the user postmaster at the sales-dept.mycompany.com department mail server.

You can use Router account-level records to reroute mail 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 mail 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, all 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:

```
<sales@client1.com> = sales-client1
client1.com = new.client1.com
```

These records route all messages 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 @ sign).

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-*
```

Mail to sales@client5.com address will be stored on your server in the cl5-sales Main Domain account, messages to info@client5.com address will be stored in the cl5-info Main Domain account, while messages to sales@client7.com address will be stored in the cl7-sales Main Domain account.

This method can be used when you do not want to create full-scale CommuniGate Pro [Domians](#) for many domains that should host 1-2 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.

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	
e164.arpa	
e164.org	

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.

Special Addresses

If the domain part of an address is `NULL`, or if the domain name part is empty, and the local part is `NULL`, the 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. This feature allows you to use a local name `NULL` or the domain `NULL` as a "black

hole" address: all messages sent to that address are just discarded. The MAILER-DAEMON address is automatically rerouted to NULL.

Example:

```
bad.company.com = null
<junk> = null
```

With these records in the Routing Table, the Server will discard all mail sent to the domain bad.company.com, as well as all mail sent to the Main Domain address junk.

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 the "Blacklisted Address" error report.

Example:

```
bad.company.com = error
<junk> = error
```

With these records entered, the Server will reject all mail sent to the domain bad.company.com, as well as all mail sent to the Main Domain address junk.

If the domain name part of an address is BlackListed, or if the domain name part is empty, and the local name part is BlackListed , the address is rejected without processing, generating the "Blacklisted Address" error report. See the [SMTP module](#) description for the details.

If the domain name part is empty, and the local name part is spamtrap, routing stops. Addresses of that type are rejected as the ERROR addresses, but the [SMTP module](#) processes them in a special manner. See the [Protection](#) section for the details.

If the domain name part ends with the symbols .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
```

Mail to all subdomains of the xyz.com domain is rerouted to the subdomains of the abc.com domain, except for mail to dept1.xyz.com and dept2.xyz.com subdomains which is 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. The suffix is removed, and the 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. The address is routed to the [SIP](#) module for signal operation, or to the [SMTP](#) module for mail transfer and access operations.

Sample 1:

The Server Main Domain name is `company.com`.

Mail for the `sales.company.com` Domain should be relayed to a separate `sales.company.com` server, while mail to all other subdomains of `company.com` should be processed as mail 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.via ; explicitly relay to a remote host
*.company.com    = company.com           ; all other subdomains are rerouted
```

You can also specify this routing using IP addresses:

```
sales.company.com = [192.0.0.1]          ; explicitly relay to the IP address
*.company.com    = company.com           ; all other subdomains are rerouted
```

Note: the addresses sent to the `sales.stalker.com` domain will be relayed with the domain part removed, i.e. the address `<user@sales.stalker.com>` will be relayed to `sales.stalker.com` host as `<user>`. This may cause troubles if the `sales.stalker.com` server does not accept addresses without domains. See the next sample for a possible solution.

Sample 2:

All mail to the domains `client1.com`, `client2`, and `client3.com` should be sent to the site `host.com`.

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 Route 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 host.

Address Processing without the `.via` suffix

<code>user @ client1.host</code>	Router converts to	<code>user%client1.host @ relay</code>
<code>user%client1.host @ relay</code>	Router converts to	<code>user%client1.host @ host.com</code>
<code>user%client1.host @ host.com</code>	Router stops	<i>no rule for host.com</i>
<code>user%client1.host @ host.com</code>	Router accepts	for SIP/SMTP <code>host.com</code> host as <code>user%client1.host@host.com</code>

Address Processing with the `.via` suffix

<code>user @ client1.host</code>	Router converts to	<code>user%client1.host @ relay</code>
<code>user%client1.host @ relay</code>	Router converts to	<code>user%client1.host @ host.com.via</code>
<code>user%client1.host @ host.com.via</code>	Router accepts	for <code>host.com</code> as <code>user%client1.host</code>

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.26.via --> host.domain:26`

When the domain name contains a colon symbol, the SIP 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 sign as the number of the TCP 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 name separator to the colon sign).

This domain name (after removal of the optional port name and its separator) is added to the local name, using the @ sign as the separator.

Sample 1:

The Server Main Domain is `company.com`.

Mail 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 mail to all other subdomains of `company.com` should be processed as mail 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
```

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 an address, the message is enqueued to this module for delivery.

Each module is called twice. First, the Router calls each module asking to process "obvious" 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 exiting 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 E-mail 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 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.

Sample 1:

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 sign 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
```

Sample Address:

```
+14153837164@some.local.domain
```

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@communicate.com is found, the sip: prefix is removed, and the Router restarts to process the pbx@communicate.com address.

Sample Address:

```
+14155551212@some.local.domain
```

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.

Default Records

When the server is first installed, the following records are placed into the Routing Table:

`<root> = postmaster`

This record reroutes all mail to the user "root" 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 domain name "localhost" is a synonym for the local IP address of this computer, 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 main server domain.

`<blacklist-admin*@blacklisted> = postmaster`

This record implements ["white hole" processing](#) for blacklisted hosts.

`<7(2d)*> = 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.

`Signal:<+(8-20d)*> = +*@telnum`

This record redirects all Signals (calls) sent to *+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#pbx

This record redirects all Signals (calls) sent to the 911 addresses in any local Domain to the emergency application started on behalf of the pbx Account in the Main Domain.

Signal:<112@*> = emergency#pbx

This record redirects all Signals (calls) sent to the 112 addresses in any local Domain to the emergency application started on behalf of the pbx Account in the Main Domain.

Signal:<(7d)@*> = localAreaCode{*}#pbx

This record redirects all Signals (calls) sent to 7-digit numbers in each local Domain to the localAreaCode 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 E-mail 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 Address	Reroute To
<input type="text" value="abuse"/>	<input type="text" value="postmaster@maindomain.dom"/>
<input type="text"/>	<input type="text"/>

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 mail to all `postmaster` names in all domains to go to the `postmaster` account in the main CommuniGate Pro domain, you should use:

```
postmaster -> anyone@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 Address	Reroute To
91*	*@domain1.dom
92*	*@domain2.dom
93*	*@domain3.dom

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 2xx number. They dial the 91 prefix (a 912xx 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

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 the Internet and your Server resources, but making mail retrieval very slow and difficult for your users.

In order to distribute their messages to millions of E-mail addresses, spammers try to use any SMTP mail server on the Internet as a relay: they deliver one copy of the message to each mail server, requesting that server to route the message to several hundred addresses. This practice not only overloads your Server resources, but it places you at risk of being recognized as a spammer (since "spam" messages come from your Server).

The CommuniGate Pro Server has Anti-Spam Options that can help you to deal with "spam".

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*.

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 mail to other places on the Internet. Messages coming from other sources should go only to your own Accounts, and should be relayed to other Internet sites only when you

explicitly allowed that type of relaying.

Specifying Client IP Addresses

The simplest way to decide if an incoming SMTP message 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

10.10.12.24 ; our back-up
10.10.13.32-10.10.13.191 ; our dial-ups
64.173.55.169 ; our other office

☒ 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 [System Administrator](#) 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.

<input checked="" type="checkbox"/>	Detect Clients by DNS Name
	<input type="text" value="*.stalker.com"/>
	<input type="text"/>

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 the 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 Settings](#) 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.

On the [SMTP Settings](#) page you can set the Relay to Clients option. Set this option to "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".

When the "two-server relay" method is used:

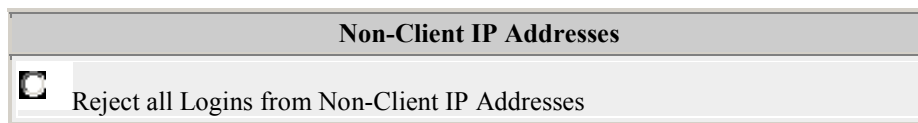
- a spammer sends a message with the `username%somehost@server2` address to the `server1` server;
- the `server1` server relays the message to the `server2` server, because the `server2` address is included into the `server1` Client IP Addresses list;
- the `server2` server relays the message to `username@somehost`, 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



If you do not plan to support [mobile users](#), you may want to select the Reject all Logins from from Non-Client IP Addresses option to allow any type of "login" operations 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, Personal File Sites, 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 Reject all Logins from from Non-Client IP Addresses option if you want to support mobile users. Select the Allow Mobile Users to Login option instead.

The SMTP AUTH method

Many E-mail clients (including Microsoft Outlook Express, Netscape Messenger, Qualcomm Eudora, and many others) now 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 mailboxes.

Non-Client IP Addresses	
	Reject all Logins from Non-Client IP Addresses
	Allow Mobile Users to Login and
<input checked="" type="checkbox"/>	Process as a Client IP Address for 30 minutes after the user disconnects
	Remember up to: 100 such addresses

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 mailbox 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 not be able to connect to that account from an internet address not 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 a user or a domain has this service disabled, the IP address from which they log in are not remembered as "temporary client IP addresses", and the SMTP Authentication will not allow those users 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) addresses 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.

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 account users.

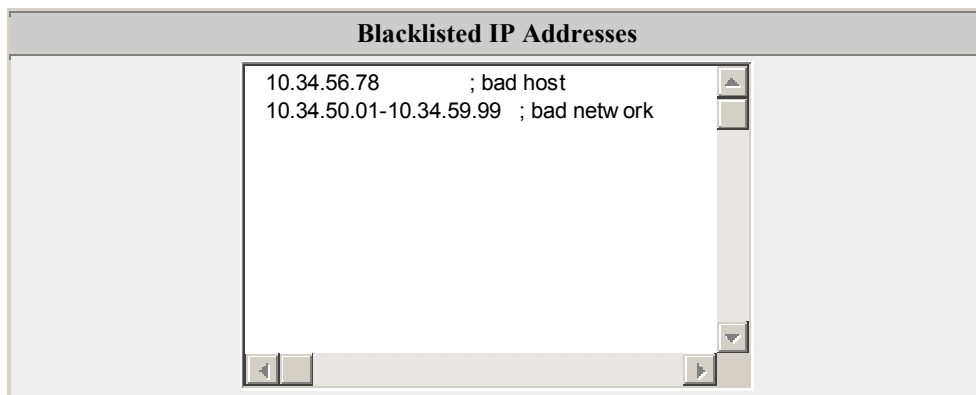
To protect your system from known spammer sites, CommuniGate Pro provides several methods to maintain "black lists" of offending hosts IP addresses.

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.

Use the WebAdmin Interface to open the Network pages inside the Settings section (realm), and click the Blacklisted IP Addresses link.

Specifying Offender Addresses

Enter the IP addresses of offending hosts in the Blacklisted IP Addresses field:



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` 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 `aaa.bbb.ccc.ddd` and this address is not listed in the Blacklisted, Unblacklistable, or Client Addresses lists, the module composes a fictitious domain name `ddd.ccc.bbb.aaa.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 `aaa.bbb.ccc.ddd` 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.

<input checked="" type="checkbox"/> Use Blacklisting DNS Servers (RBLs)
<div> <input type="text" value="rbl.rblprovider.com"/> </div> <div> <input type="text"/> </div>

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 not 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 Client IP Addresses and 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.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 address is processed as a blacklisted one.

<input checked="" type="checkbox"/> Blacklist by DNS Name
<div> <input type="text" value="*.dial-up.telco.dom"/> </div> <div> <input type="text"/> </div>

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:

<input checked="" type="checkbox"/> Blacklist by DNS Name
<input type="text" value="*.dial-up.telco.dom"/>
<input type="text" value="(host name is unknown)"/>
<input type="text"/>

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

192.168.254.18 ; a good fellow , got into RBL by mistake

Note: Addresses listed in your Client IP Addresses list are never checked using any Blacklisting method, so there is no reason to include the Client IP Addresses into the UnBlacklistable IP Addresses table once again.

You can "unblacklist" addresses using their DNS (PTR) names:

<input checked="" type="checkbox"/>	UnBlacklist by DNS Name
	<input type="text" value="*.stalker.com"/>
	<input type="text"/>

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	
<input type="radio"/> Reject	<input type="radio"/> Add Header: <input type="text" value="X-Spam: ^1 blacklisted(^0)"/>

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.

Automatic Blacklistings

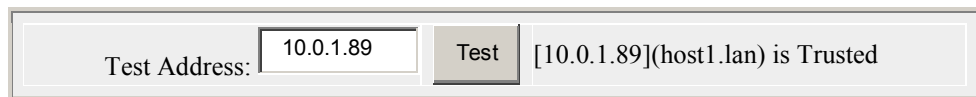
CommuniGate Pro maintains its own "temporary Blacklist". Addresses in that list are blacklisted for the speci-

fied period of time only. See the [SMTP](#) module description for more details.

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:



The screenshot shows a web interface for testing network addresses. It consists of a light gray rectangular box with a thin border. Inside, on the left, is the text 'Test Address:' followed by a text input field containing '10.0.1.89'. To the right of the input field is a gray button with the text 'Test'. Further to the right, the result of the test is displayed: '[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
```

You do not have to create fictitious accounts, you should create the Router alias records only.

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.

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 page in the Settings section of the WebAdmin Interface, and click the RFC822 Receiver link.

Banned Header Lines	
	Subject: Body Fat Loss-No Dieting!
	From: The Legal Network <*@*hackedisp.com>

Banned Body Lines	
	No special skills or experience is required. We will give you










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 or empty **To :** and/or **Cc :** header fields:

Data	Operation	Parameter
Header Field 	is not 	To:*
Action	Parameters	
Reject with 	<div> Messages without To: fields are  </div> <div>  </div> <div>  </div> <div>    </div>	

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 or just use a record without any 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 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 `bigprovdiar.com` to be sent via a particular relay `relay3.com`, you should place the following record into the Router table:

```
NoRelay:bigprovdiar.com = bigprovdiar.com@relay3.com.via
```

Without the `NoRelay` prefix, any host on the Internet could send messages to `bigprovdiar.com` via your Server. The `NoRelay` prefix tells the Router not to add marker to addresses in the `bigprovdiar.com` domain, so only your own users (clients) can send mail to `bigprovdiar.com` domain using your Server.

Note: you may have an alias record in your Router:

```
<joe> = joe5@bigprovdiar.com
```

This record tells the server to reroute all mail addressed to `joe@mydomain.com` to `joe5@bigprovdiar.com`. Since this record has the (default) `Relay:` prefix, anybody in the world can send messages to `joe@mydomain.com` and those messages will be successfully relayed to the `bigprovdiar.com` domain. The `joe5@bigprovdiar.com` address will be converted to `joe5%bigprovdiar.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	<code>joe@mydomain.com</code>	NO
Main Domain (<code>mydomain.com</code>) cut-off:	<code>joe</code>	NO

Router Record: <joe> = joe5@bigprovdier.com	joe5@bigprovdier.com	YES
Router Record: NoRelay:bigprovdier.com = big- provdier.com@relay3.com.via	joe5%big- provdier.com@relay3.com.via	YES
SMTP Module: 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 IP domains" (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 "blacklisted IP domains" (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.



Security

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 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).

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 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 Server supports the following GSSAPI authentication methods:

- Kerberos V5

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 [WebUser_authentication](#) method.

Use the WebAdmin Interface to open a [Domain Settings](#) page and find the Login Methods panel:

Login Methods																	
<input type="checkbox"/>	default	<input checked="" type="checkbox"/>	CLRTXT	<input checked="" type="checkbox"/>	CRAM-MD5	<input checked="" type="checkbox"/>	DIGEST-MD5	<input checked="" type="checkbox"/>	APOP	<input type="checkbox"/>	GSSAPI	<input type="checkbox"/>	NTLM	<input checked="" type="checkbox"/>	MSN	<input type="checkbox"/>	WEBUSER

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 than 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.

WebUser

This option enables the [WebUser](#) 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.

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 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, 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-crypt` Password Encryption option is selected, an enhanced Blowfish-based encryption is used.

`U-crypt` and `UB-crypt` 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 than "on-the-disk" security (using one-way password encryption methods).

`U-crypt` 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 non-crypted 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	<p>The Windows NT domain authentication system is used. The Windows account used to run the CommuniGate Pro Messaging Server should have the <code>Act as part of the operating system privilege</code>.</p> <p>The <code>--BatchLogon</code> command line option can be used to tell the Server to use the <code>LOGON_BATCH</code> authentication method (if the option is not present, the <code>LOGON_NETWORK</code> method is used).</p> <p>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. If Accounts in the <code>company1.dom</code> CommuniGate Pro domain have the OS User Name setting set to <code>*%comp1</code>, then the OS user name for the CommuniGate Pro Account <code>joe</code> will be <code>joe%comp1</code>, and the CommuniGate Pro Server will use the Windows <code>LogonUser</code> API to try to authenticate the mail client as the Windows user <code>joe</code> in the Windows domain <code>comp1</code>.</p>
Unix-based systems	The <code>passwd</code> and <code>shadow</code> or other OS-supported authentication mechanisms are used.
OS/400 systems	The <code>user profile</code> 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 Ker-

beros links. The list of Domain Kerberos Keys will be displayed:

	Realm	Principal	Issued	Version	Type
<input type="checkbox"/>	REALM1.COM	(3)imap/domain1.com	19-05-2004 17:36:33	6	RC4-HMAC
<input type="checkbox"/>	REALM1.COM	(3)imap/domain1.com	21-05-2004 17:32:35	9	DES-MD5
<input type="checkbox"/>	REALM1.COM	(3)imap/domain1.com	24-05-2004 12:41:55	10	DES-MD5
<input type="checkbox"/>	REALM1.COM	(3)imap/domain1.com	24-05-2004 17:33:27	13	DES-CRC32
Delete Marked		Import Keys:			

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: The Kerberos Authentication mechanism does NOT run the resulting user name via the Router. If it would use the Router, a name in one Domain could be routed to a name in a different Domain, so the Kerberos Keys valid for one Domain would provide access to Accounts in a different Domain. As a side effect, the Account

Aliases cannot be used in Kerberos tickets.

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 by the Trusted Certificate selected on the Server.

If the client presents such a Certificate, the E-mail field of the Certificate Subject is interpreted as the name of the CommuniGate Pro Account to log into.

Note: The Certificate Authentication mechanism does NOT run the provided E-mail address via the Router. If it would use the Router, a name in one Domain could be routed to a name in a different Domain, so the Kerberos Keys valid for one Domain would provide access to Accounts in a different Domain. As a side effect, the Account Aliases cannot be used with Certificate Authentication.

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, and/or to assist with the [Router](#) operations.

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:

<input checked="" type="checkbox"/> External Authentication			
Log:	Major & Failures	Program Path:	/usr/sbin/pasw check
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 [Helpers](#) section to learn about the External Authentication interface protocol.

Sample External Authentication programs and scripts can be found at the <http://www.stalker.com/CGAUTH/> 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.

To prevent this type of attack, you may want to enable the `Hide Unknown Account messages` option, located on the `Obscure` page in the WebAdmin Settings realm:

Login Security	
<input checked="" type="checkbox"/>	Hide Unknown Account messages
Suspend Account after	5 failed logins within 2 minutes

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.

Account Password Attacks

The CommuniGate Pro server can temporarily disable all types of login operation for an Account that has seen too many incorrect login attempts. The Login Security panel shown above allows you to 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 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 you should have the "Can Modify Access Rights" 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 Modify Access Rights (unlimited access):`

This setting specifies if the user is allowed to modify Access Rights of CommuniGate Pro users. If some users are granted this right, they can access all Server settings and pages (i.e., all other rights are granted, too).

`Can Modify User Accounts`

This setting specifies if the user is allowed to create, remove and delete Accounts and Domains, and to change Account and Domain Settings.

`Can Modify Server Settings`

This setting specifies if the user is allowed to change configurations of CommuniGate Pro services (SMTP, POP, Router, etc.).

`Can Monitor Server`

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 Unlimited Access right.

Select the desired Access Rights and click the Update button.

The Access Rights are stored in one file for each domain, the Access.settings file stored in the Settings subdirectory of the domain directory. This makes it easy to check to whom the Server administration rights are granted.

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 Protection page:

`Reject all Logins from Non-Client Addresses`

When this option is selected, all "login" operations (needed for POP, IMAP, 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](#) and access to [Personal File](#)

Sites.

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.

Note: Before you enable this option, make sure that the address you are using is included into the Client Addresses list: otherwise you will immediately lose 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 Authorisation Account) is being opened.

Impersonating is supported for PLAN and GSSAPI Authentication methods. It can also be used for [Real-Time Registration](#).

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](#).

Using the WebUser Authentication Method

The CommuniGate Pro Server supports the special `WebUser` authentication method. This method uses the session ID of a [WebUser](#) session instead of the account password. The method is useful for [CGI](#) programs and scripts. This method is disabled by default (see above).

The WebUser SASL method works only for programs running on the same Server computer, or for programs running on other servers in the CommuniGate Pro [Cluster](#).

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 WebUser session ID.

The WebUser authentication operation for the PWD module is:

```
AUTH WEBUSER userName session-ID
```

The WebUser authentication operation for the IMAP module is:

```
AUTH WEBUSER 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 WEBUSER john@doe.dom 114-bXaKw92JK1pZVB5taj1r
```

opens the john@doe.dom account in this PWD session.



Public Key Infrastructure

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 possesses 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"

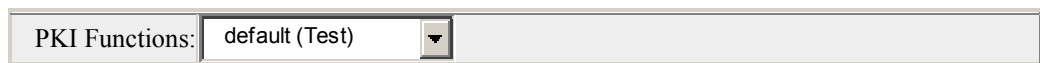
Multiparty Encryption

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:



The image shows a web interface element for PKI Functions. It consists of a label 'PKI Functions:' followed by a dropdown menu. The dropdown menu is currently open, showing the selected option 'default (Test)' and a small downward arrow icon. The background of the dropdown is light gray.

This option allows you to specify the PKI Mode for this Domain,

No

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 *Stalker Software, Inc.* as the Issuer. The Test Certificate expires 30 days after the last Server restart time.

Yes

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.

A screenshot of a web interface for assigning a private key. It features a title bar labeled "Private Key". Below the title bar, there is a button labeled "Generate Key" and a label "Size:" followed by a dropdown menu currently showing "512 bit".

Private Key	
Generate Key	Size: 512 bit ▼

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 (such as OpenSSL) to generate a Private Key. You should instruct that program to output the Private Key in the `PEM` format (as shown below). Select "Custom" in the Size: menu and click the Generate Key button. A text field appears. Copy the PEM-encoded Key into that text field, and click the Generate Key button:

Private Key

Generate Key Size: Custom

Enter a PEM-encoded Private Key:

```
-----BEGIN RSA PRIVATE KEY-----
MIIBPAIBAAJBBAKrbegkuRk8VcRmWFmtP+LviMB3+6dizWW3Dw affz
Tv0XtbsCyl3QoyKGhrOAY3RvPK5M38iuXT0CAw EAAQJAZ3cnzaH
rD1qFBAVfoQFIOH9uPJgMaoAuoQEIsPHVcZDKcOv4w Eg6/TlnAIXB
oQlhAPcgZzUq3yVooAaoov8UbXPxqHlw o6GBMqnv20xzkf6ZAIeA
mvpCPHZw QJdmdHHkGKAs37Dfxi67HbkUCIQCeZGllHXFa071Fp06Ze
rJBhdTe0v5pCeQlhAIZfkiGgGBX4cluuckzEm43g9WMUjxP/0GIK39vly
-----
```

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+Or3zXm1N7ILW1YDsR6cgPlzHomAxi6aeUthl4lSqBHaqMlh+/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: 1024 bit Remove Key & Certificate

Key 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 Ke, 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 . com-pany . 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

Common Name:

d1.stalker.com

Country:

US

Province/State:

CA

City/Town:

Sausalito

Organization Name:

ACME Yacht Rentals, Inc.

Organization Unit:

On-line Services

Contact E-mail:

bill@domain.company

Generate Self-Signed

Create Signing Request

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 E-mail

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.

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.

You can Click the Generate Self-Signed button when a Self-Signed Certificate is already assigned. In this case the Server creates a new Self-Signed Certificate with the same serial number, but with new validity period.

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

Common Name:

d1.stalker.com

Country:

US

Province/State:

CA

City/Town:

Sausalito

Organization Name:

ACME Yacht Rentals, Inc.

Organization Unit:

On-line Services

Contact E-mail:

bill@domain.company

Generate Self-Signed

Create Signing Request

Certificate Signing Request (CSR)

-----BEGIN CERTIFICATE REQUEST-----
MIIBZTCCAQ8CAQAw gaxxCzAJBgNVBAYTAiVTMQsw CQYDVQQQ
BxMJU2F1c2FsaXRvMSEw Hw YDVQQKEzhBQ01FIFhY2h0IFJlbnRl
GTAXBgNVBAU0TEE9uLWxpbnUgU2VydmljZXN0FzAVBgNVBAMTLC
Y29tMSw IAYJKoZIhvcNAQkBFhNiaWxsQGRvbWVpbi5jb21w YW5l
hvcNAQEBBQADSw Aw SAJBAKrbekuRk8VcRmWFmtP+LviMB3+6
GAFw UJ/ITv0XtbsCyl3QoyKGhrOAY3RvPK5M38iuXT0CAw EAAaA/
DQEBBAUAA0EABFysKDutO/ZF16Sp7+LL3lcjUb29xivZSnBXYsD/q
4Qyxcio6ynsh7HNjdJBfdFyO+J2yw XZiIYg==
-----END CERTIFICATE REQUEST-----

submit this request to a Certification Authority and paste the result below

Enter a PEM-encoded Certificate

Set Certificate

----- S Y S T E M -----

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:

Certificate	
Issued to:	Country: US Province: CA City: Sausalito Organization: ACME Yacht Rentals, Inc. Unit: On-line Services Common Name: d1.stalker.com Contact: bill@domain.company
Issued by:	Country: ZA Province: FOR TESTING PURPOSES ONLY Organization: Thawte Certification Unit: TEST TEST TEST Common Name: Thawte Test CA Root
Serial Number: 89AB673940123456	
Valid:	From: 06-Jun-05 Till: 06-Jan-07
<input type="button" value="Remove Certificate"/>	

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 Secure Certificate To Use option is set to Yes.

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:



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 soft-

ware - 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 Domian. 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	Valid From Valid Till
Organization: VeriSign Trust Network Unit: VeriSign, Inc. Unit: VeriSign International Server CA - Class 3 Unit: www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.(c)97 VeriSign		Country: US Organization: VeriSign, Inc. Unit: Class 3 Public Primary Certification Authority	16-Apr-97 07-Jan-04
		Remove CA Chain	

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

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.

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

To update the set of Server-wide and Cluster-wide Trusted Certificates, open the Domains section of the WebAdmin Interface and follow the Security link.

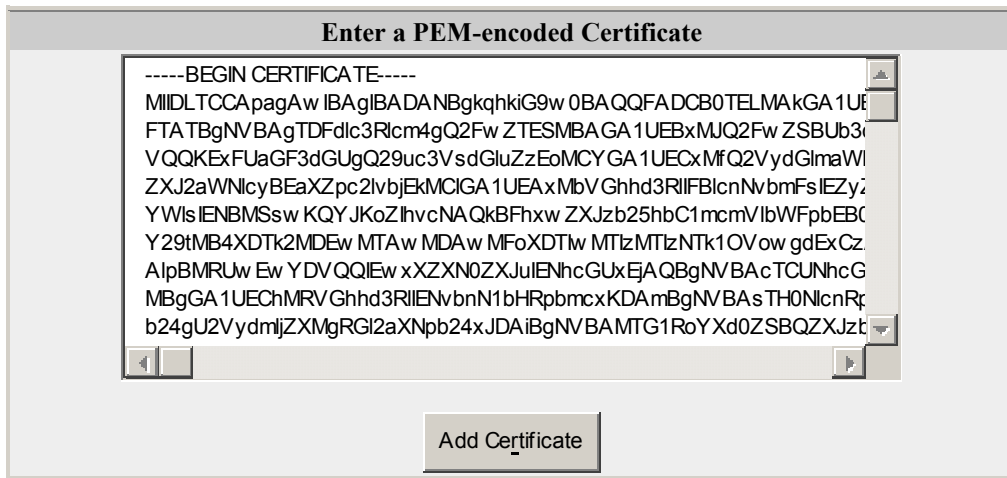
To update the set of the Domain-wide Trusted Certificates, use the WebAdmin Interface to open the Domain Settings page of the target Domain, open the Domain Security pages and follow the Trusted link.

The Trusted Certificates page will open:

	Subject	Serial Number	Valid From	Valid Till
	SecureSign RootCA3	5F60585F00000000	15-Sep-99	15-Sep-20
	Thawte Personal Freemail CA	00	31-Dec-95	31-Dec-20
	Thawte Personal Freemail CA	00	31-Dec-95	31-Dec-20
<input type="checkbox"/>	My Company CA	010256FF	31-Dec-04	31-Dec-20
<div>Remove Marked</div>				

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.



To add a Certificate to the set, enter the PEM-encoded Certificate data into the text field and click the Add Certificate button. The new Certificate should appear in the displayed set.

SSL/TLS Secure Connections

The TLS (Transport Level Security) protocol is 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 mod-

ules: 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.

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.

To specify the Server-wide SSL/TLS processing parameters, open the Obscure page in the Settings section of the WebAdmin Interface:

TLS Sessions	
Log:	Major & Failures
Time To Live:	30 seconds
<input type="checkbox"/> Weak Ciphers	

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.

Weak Ciphers

Select this setting if you want to support weak (less than 128-bit) security (cipher methods).

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:	<input type="text" value="Certificate Name 1"/>

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. If a client sends such a certificate, the E-mail field from the certificate Subject can be used for the Certificate-based Authentication.

Note: the E-mail address specified in the Certificate is used "as is", it is not processed with any [Router](#) mechanism.

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 Interface supports S/MIME functions. It offers a secure storage for user Private Key. If a desktop client application is used instead of the WebUser Interface, the user Private Key should be stored in the PKI storage of the desktop operating system. The WebUser Interface can export and import Private Keys, so the user can use the same Private Key for desktop applications and for the WebUser Interface. 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 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 jon and jim Certificates, they both of them can decrypt this message.

Stored Messages Encryption

After CommuniGate Pro users receive certain clear-text E-mail messages, they may prefer to keep them encrypted in the Server mailboxes. The [Web User](#) Interface provides the Encrypt and Decrypt functions that allow users to encrypt and decrypt individual messages in their mailboxes.



Lawful Interception

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, as well as all message manipulation activity in those Accounts is reported via E-mail messages sent to the 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 local law enforcement agencies.

Configuring Interception Settings

To configure the Interception settings, open the General page in the Settings realm of the WebAdmin Interface. Click the Intercept link to open the Interception Settings page:

Account Name	Send Reports to	Login	New Mail	Sent	Mailbox Access	Partial
john.the.ripper@domain1.dom	Sgt.Smith <joe.smith@fbi.gov>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
jim.the.dripper@domain2.dom		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

To add an element to list, fill the Account Name field in the last empty row and specify the E-mail address to send the reports to, select the checkboxes for the reports you want 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.

Report Message Formats

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 the user logged in from
- The Protocol used (POP, IMAP, WebUser, etc.)

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 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 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 Mailbox name and, for the rename operation, the new Mailbox name.

Access Report

The Access report is sent when a monitored user reads or removes 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 Mailbox name.
- The UID(s) of the message(s) read or deleted. For partial access reports, the message portion specification is included, too.
- The Protocol used (POP, IMAP, WebUser, etc.)

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 network (IP) address the user logged in from
- The name of authenticated user (if any); (`self`) means the monitored user name.
- The Protocol used (POP, IMAP, WebUser, etc.)

The second part is an `application/x-envelope` part and contains a copy of the message envelope.

lope data.

The third part is a `message/rfc822` part and contains a copy of the submitted message.



Scalability

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 (2,000 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 high-end mail servers may want to disable the Use Conservative Info Updates option (located in the Local Account Options on the WebAdmin Obscure Settings page). This 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 $100,000 * 2 * 15 \text{ sec} / (12 * 60 * 60 \text{ sec}) = \sim 70$ 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.

If supporting 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.

SIP/RTP Clients

As compared to messaging, which tends to be very disk I/O-limited, SIP and RTP data has real-time requirements and only a few actions (such as a SIP REGISTER) cause any disk I/O. Real-time traffic is highly susceptible to any network or system latency, and as such is more closely tied to CPU performance than email. 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 signalling 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 [RealTime](#) 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 OSs 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 NT 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.

HyperThreading

Most operating systems (including Linux, FreeBSD, Solaris, Windows) do not fully support x86 HyperThreading.

Unlike multi-core and/or multi-CPU technologies, HyperThreading provides very small (if any) performance gain, but it causes many problems: threads library crashing, poor NFS performance under load, etc.

Use your server BIOS settings to disable HyperThreading.

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.

You may want to try various values for the Concurrent Requests setting in the Domain Name Resolver panel on the Obscure Settings page: 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 Resources 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 (BSD-type) 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/STLKCPro.init` to be executed at boot time. The following is a Startup.sh file for CommuniGate Pro version 4.3 or later for most Solaris 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::

```
/var/CommuniGate/Startup.sh
### Startup.sh begin
SUPPLPARAMS="--DefaultStackSize 131072 --useNonBlockingSockets --closeS-
tuckSockets --CreateTempFilesDirectly 10"
ulimit -n 16384
### Startup.sh end
```

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.

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.

Additions to /etc/system:

```
* 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
nnd -set /dev/tcp tcp_recv_hiwat 49152 # or 65536
nnd -set /dev/tcp tcp_xmit_hiwat 49152 # or 65536
# increase the connection queue
nnd -set /dev/tcp tcp_conn_req_max_q 512
nnd -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

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](https://support.microsoft.com/kb/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:

```
/var/CommuniGate/Startup.sh
### Startup.sh begin
SUPPLPARAMS="--DefaultStackSize 131072 --useNonBlockingSockets --closeS-
tuckSockets --CreateTempFilesDirectly 10"
ulimit -n 16384
### Startup.sh end
```

Linux kernel 2.2 and previous:

The pre 2.2.x Linux kernels allowed a process to open 256 files descriptors only. If you want your server to handle more than 100 TCP/IP connections, use the Linux kernel 2.2.x or a later version to avoid the "out of file descriptors" problem.

The Linux threads library uses the one-to-one model, so each CommuniGate Pro thread is a kernel thread (actually, a "process"). This may be not the best solution for very large systems that should run several thousand threads.

In spite of the fact that the Linux threads are handled within the kernel, the Linux thread library has its own scheduler, too. By default, that scheduler uses a static table with 1024 entries, so no more than 1024 threads can be created. This is enough for even large sites serving may POP and WebMail users, but can cause problems for sites that need to serve several hundred concurrent IMAP users. To increase this number, the Linux threads library has to be recompiled with the `PTHREAD_THREADS_MAX` parameter increased.

The Linux threads library allocates thread stacks with 2MB steps. This does not allow the system to start more than 1000 threads on 32-bit machines. CommuniGate Pro threads do not need stacks of that size. You may want to recompile the Linux threads library decreasing the `STACK_SIZE` parameter to 128K.

Linux kernel 2.4:

Linux kernel 2.4 allows for most important tuning options to be easily configured. In some 2.4 distributions, however, `PTHREAD_THREADS_MAX` may still be compiled at 1024 or another similarly low number - if this is the case, the Linux threads library must be recompiled with the `PTHREAD_THREADS_MAX` parameter increased. Most Linux 2.4 distributions are based on the "LinuxThreads" threading library. There are some Linux 2.4 distributions which have attempted to back-port the newer POSIX threading library to the 2.4 kernel - in some instances, running POSIX threads with 2.4 kernel has proven to cause stability problems for many applications, including CommuniGate Pro. It is recommended to use the LinuxThreads library with CommuniGate Pro when running the 2.4 kernel. This is provided for (by default) in the `/etc/init.d/CommuniGate` startup script:

```
LD_ASSUME_KERNEL=2.4.1
export LD_ASSUME_KERNEL
```

Following are some additional tuning options for Linux 2.4. 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 2.4 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
```

One other note about kernel 2.4 and 2.6 - there is a noticeable networking performance improvement in compiling your kernel without NETFILTER (iptables) if not using iptables on your CommuniGate Pro server.

Linux kernel 2.6 and later:

Linux kernel 2.6 introduced "POSIX threads", replacing the previous default thread library "LinuxThreads". Kernel 2.6 implementations are the first Linux release for which using POSIX threading is recommended. If using Linux kernel 2.6 and POSIX threading (and preferably using a distribution for which kernel 2.6 was the default kernel), then the `/etc/init.d/CommuniGate` init script should be modified to comment out the following lines:

```
# LD_ASSUME_KERNEL=2.4.1
# export LD_ASSUME_KERNEL
```

Commenting these lines should allow for the POSIX thread shared libraries (located in `/lib/tls/`) to be linked in by default.

Following are some additional 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 2.4 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 both FreeBSD 4 and FreeBSD 5.

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::

```
/var/CommuniGate/Startup.sh
### Startup.sh begin
SUPPLPARAMS="--DefaultStackSize 131072 --useNonBlockingSockets --closeS-
tuckSockets --CreateTempFilesDirectly 10"
ulimit -n 16384
### Startup.sh end
```

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"
```

Building an optimized kernel allows for an increase in maximum process size and increasing the number of nmb-

clusters for networking capacity:

```
# increase max allowable process virtual memory size
options          DFLDSIZ="(128*1024*1024) "
options          MAXDSIZ="(2048*1024*1024) "
options          MAXSSIZ="(2048*1024*1024) "
# increase nmbclusters
options          NMBCLUSTERS=65536
```

FreeBSD 4

While FreeBSD 4 is renown as an incredibly stable platform, it does not provide Symmetric Multiprocess (SMP) support for multithreaded applications. Because of this, it typically does not make cost-effective sense to allocate anything more than a single CPU server with FreeBSD 4, except perhaps in the case where there are multiple applications (in addition to CommuniGate Pro) running on the system, such as perhaps a DNS server on the system used for faster name service queries and caching.

Following are some additional tuning options for FreeBSD 4. Generally, this script should be located at `/usr/local/etc/rc.d/tuning.sh`:

```
#!/bin/sh
# FreeBSD 4 tuning script
# increase socket buffers
sysctl -w kern.ipc.maxsockbuf=1048576
sysctl -w kern.ipc.somaxconn=4096
# increase default buffer size
sysctl -w net.inet.tcp.sendspace=65536
sysctl -w net.inet.tcp.recvspace=65536
# decrease time wait
sysctl -w net.inet.tcp.keepidle=300000
sysctl -w net.inet.tcp.keepintvl=30000
# increase vnodes
sysctl -w kern.maxvnodes=131072
# increase maxfiles/maxfiles per process
sysctl -w kern.maxfiles=131072
sysctl -w kern.maxfilesperproc=32768 # or 65536 if necessary
```



```
# increase port range
sysctl -w net.inet.ip.portrange.last=65535
# default: net.inet.ip.rtxpire: 3600
sysctl -w net.inet.ip.rtxpire=300
# decrease MSL from 30000
sysctl -w net.inet.tcp.msl=10000
# vfs.vmiodirenable: cache directory locations in memory (?)
# this may help in some cases, but be careful and test if used
# sysctl -w vfs.vmiodirenable=1
```

FreeBSD 5

FreeBSD 5 introduced true SMP support for multiple CPUs, and therefore is better optimized for CommuniGate Pro's highly threaded resource needs. Sysctl settings in FreeBSD 5 can be set automatically in the `/etc/sysctl.conf` file:

```
# 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=32768 # or 65536 if necessary
# increase port range
net.inet.ip.portrange.last=65535
# default: net.inet.ip.rtxpire: 3600
net.inet.ip.rtxpire=300
```

```
# decrease MSL from 30000
net.inet.tcp.msl=10000
# vfs.vmiodirenable: cache directory locations in memory (?)
# this may help in some cases, but be careful and test if used
# vfs.vmiodirenable=1
```

HP-UX

HP-UX kernel parameters for HP-UX are set through a few different mechanisms, depending on your HP-UX version. 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
maxssiz	Maximum size of the stack 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)
maxssiz       (2048*1024*1024)
nfile         32768
ninode        32768
```

Decreasing the TIME_WAIT for TCP is also recommended:

```
ndd -set /dev/tcp tcp_time_wait_interval 60000
```

Mac OS X

In older versions of Mac OS X, there may be a 1600-1800 descriptors/process "hard limit" set by default. In CommuniGate Pro version 5.0, the maximum number of file descriptors is statically set at 8192 FDs.

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 2,000 users, and you should increase that limit by specifying the following in the CommuniGate Pro Startup.sh file:

```
ulimit -d 1048576
```

```
ulimit -n 16384
```




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](#) 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-2001	Server will be shut down on Aug,3 from 1:00pm till 1:30pm
<input type="checkbox"/>	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
<input type="checkbox"/>	23:54:10	The IMAP service is now available
<input type="button" value="Remove Marked"/>		
<input type="button" value="Post Alert"/> <input type="text"/>		

The Alerts page for a CommuniGate Pro Domain contains both Server-wide and Domain-wide alerts. The Server-wide Alerts have highlighted (bold) time stamps, they cannot be removed from this 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 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.



Triggers

The CommuniGate Pro Server can detect certain situations and process them as Triggers, invoking *Trigger Handlers*. Trigger Handlers notify System Administrators.

The Trigger Manager monitors the values of selected [SNMP](#) 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 generated when there are more than 10,000 messages in the Server Queue (the value of the `numQueuedMessages` INTEGER-type SNMP 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 SNMP Element increased for more than 5000 during the last 10 seconds).

Configuring Trigger Handlers

An Trigger Handlers specifies how a system administrator or system operator should be notified. Several notification methods are supported, and each Event Handler can use any number of supported methods.

To configure the Trigger Handlers, open the General page in the Settings realm of the CommuniGate Pro WebAdmin Interface, and follow the Events link. The list of existing Trigger Handlers will appear. Each Trigger Handler has a name and the notification method parameters:

Handler Name:	Warning
.....methods....	

There is an empty 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 Email option:

<input checked="" type="checkbox"/> Send Email	Subject: [!] System Warning: ^0 To: postmaster, Text: The CommuniGate Pro server parameter ^0 value = ^2, threshold = ^1
--	--

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 SNMP Element that generated the Event.

^1

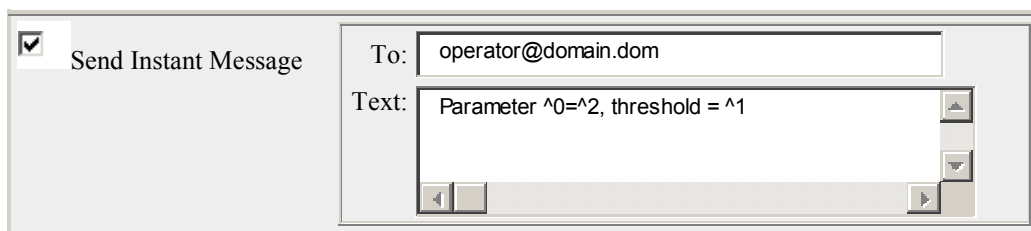
This combination is replaced with the value of the SNMP Element threshold.

^2

This combination is replaced with the actual value of the SNMP Element.

Notification via Instant Messages

To send Trigger Notifications as an Instant Message, select the Send Instant Message option:



The screenshot shows a configuration window for sending instant messages. On the left, there is a checkbox labeled 'Send Instant Message' which is checked. On the right, there are two text input fields. The 'To:' field contains the email address 'operator@domain.dom'. The 'Text:' field contains the text 'Parameter ^0=^2, threshold = ^1'. Below the 'Text:' field, there are two small buttons: a left-pointing arrow and a right-pointing arrow.

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:

<input checked="" type="checkbox"/> Send SNMP Trap	To: 10.1.2.1, remote.site.com:8162
--	------------------------------------

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 an [Account Alert](#), select the Send Account Alert option:

<input checked="" type="checkbox"/> Send Account Alert	To: operator@example.com
	Text: ^0 value is ^2

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 as a URL request, select the Send URL option:

<input checked="" type="checkbox"/> Send URL	To: <input type="text" value="http://service.example.com/notify.pl?Parm=^0&Value"/>
--	---

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 that limit the amount of event 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	<input type="text" value="3"/>	▼	alerts	within	<input type="text" value="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 generated events are still recorded in the CommuniGate Pro Log.

Specifying Triggers

You can specify an Trigger by setting a "threshold" for some [SNMP 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 a Web browser to open the Triggers page in the Settings realm of the WebAdmin Interface. Click the Elements link to specify Triggers:

Element	Handler	Threshold
smtpInputActive	Warning	1000
smtpInputTotal	---	---
smtpInputJobs	---	10000 in minute
smtpInputMessagesReceived	---	--- in ---

The Handler field specifies which Trigger Handler should be used to process this Trigger.

To remove an Trigger, reset its Threshold value (set it to ---).

If you want to disable an Event without removing its threshold value, reset its Handler field (set it to ---).



Network

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.

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 a web browser to open the Settings realm of the CommuniGate Pro WebAdmin Interface, and click the Network link. The LAN IPs page appears.

LAN IP Addresses	
10.0.0.1-10.0.0.255	; office LAN
10.0.1.12	; remote office on VPN
10.0.1.15-10.0.1.220	; DHCP LAN addresses

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.

Each table line should include either one IP address or an address range - two IP addresses separated with the minus sign: a range includes both IP addresses and all addresses between them.

A comment (separated with the semicolon (;) symbol) can be placed at the end of a line. A line starting with a semicolon symbol is a comment line.

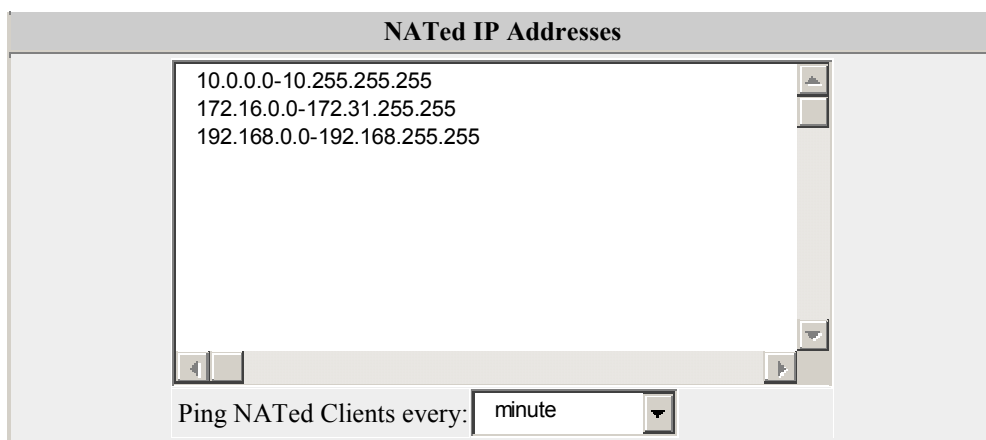
Usually, you want all mail clients connecting from the LAN addresses to be able to relay mail to any Internet destination, so you will include the LAN addresses into the [Client IP Addresses](#) list.

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 "not-real" ("local") addresses, i.e. which addresses cannot be contacted directly from the Internet.

NATed Addresses

CommuniGate Pro can provide [SIP](#) and real-time communications for remote clients located behind NAT devices, implementing the *far-end NAT traversal* functionality.

To detect clients located behind NATs, the Server needs to know which addresses are used on remote networks behind those NATs. Use a web browser to open the Settings realm of the CommuniGate Pro WebAdmin Interface, and click the Network link. Then open NATed IPs page.



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 listed included into the NATed IP Addresses list, the Server decides that this client is behind a NAT.

To allow other users to make incoming calls to a client behind a NAT, the CommuniGate Pro server keeps the "communication hole" between the client and server open by periodically sending dummy packets to that client. Use the Ping NATed Clients setting to specify how often the Server should send those packets.

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:	64.173.55.161
WAN IPv6 Address:	2001:470:1f01:2565::a:7

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, 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:

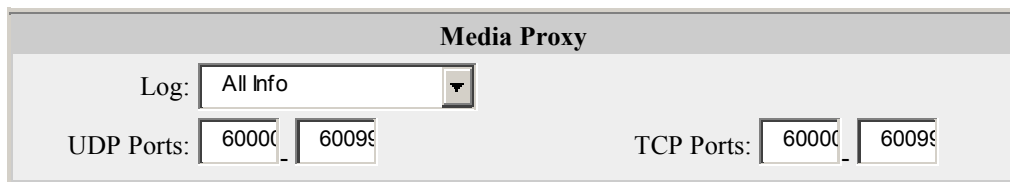
WAN IPv6 Address

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.

Media Proxy Parameters

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 real-time client on a LAN tries to communicate with a remote system on the Internet, CommuniGate Pro creates a communication port on its own system, and forces the client to connect to that port instead of the remote system port. The CommuniGate Pro communicates with the remote system itself, relaying the data received from the remote system to the client on the LAN and vice versa.

Media Proxy is used when serving real-time clients located behind [remote NAT](#) devices.



The image shows a configuration window titled "Media Proxy". It contains three main settings:

- Log:** A dropdown menu currently set to "All Info".
- UDP Ports:** Two input fields showing a range from "60000" to "60099".
- TCP Ports:** Two input fields showing a range from "60000" to "60099".

Log

Use this setting to specify what kind of information the 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 Proxy component records in the System Log are marked with the `UDPPROXY` or the `TCPProxy` tag.

UDP Ports

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 TCP 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.

Note: the [FTP Module](#) uses the ports from the TCP Ports set for Passive Mode transfers.

Domain Name Resolver (DNR)

The CommuniGate Pro Server uses its own high-speed multithreaded Domain Name Resolver to convert domain names into network (IP) addresses. To convert names, the Domain Name Resolver sends requests to the specified Domain Name Servers.

Server Administrators with the Can Modify Settings access right can modify the Resolver settings. Open the

Obscure page in the Settings section of the Server WebAdmin Interface:

Domain Name Resolver		Log: Problems
DNS Addresses:	OS-specified	[209.1.58.247], [206.40.74.1]
Initial Time-out:	7 seconds	Retry Limit: 5
Concurrent Requests:	10	

Log

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.

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 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.

Initial Time-out

The Domain Name System uses the connectionless UDP protocol by default, and if there any network trouble, a UDP request or reposnse 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.

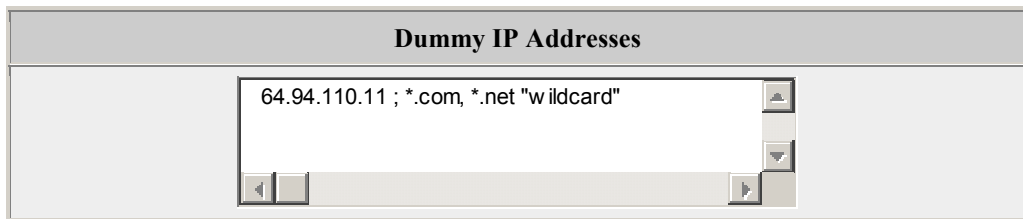
If you have several Domain Name Servers specified, each time the resolver needs to repeat a request, it sends it to the next DNS server in the list.

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 Resolvers sends the same request not more than twice, and both times it uses the same (Initial) response time-out.

The Domain Name Resolver uses TCP connections if the server UDP response came back with the "Truncated" flag set. This feature allows the Resolver to retrieve very large records from DNS servers.



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.

IPv6 Support

The CommuniGate Pro Server provides full support for both IPv4 and IPv6 network protocols for the following Server Operating Systems:

- Solaris
- FreeBSD
- NetBSD
- MacOS X
- Linux
- 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.

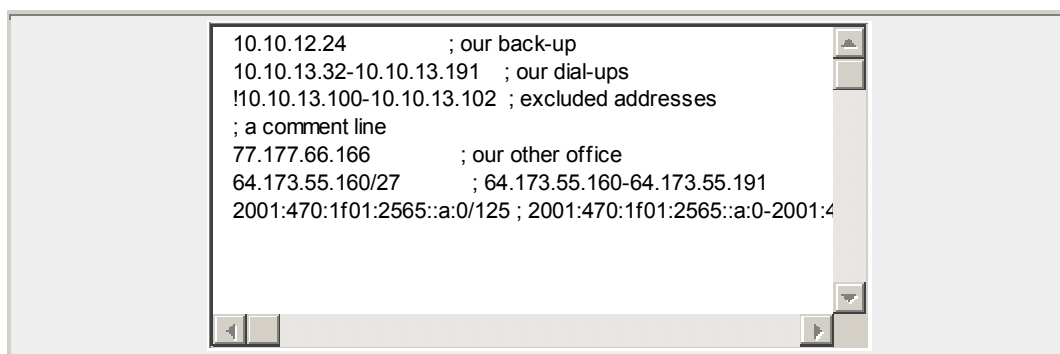
You can explicitly instruct the Server to switch IPv6 networking support on or off by using the `--IPv6` [Command Line Option](#):

- `--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.

Network Address Lists

Many CommuniGate Pro components use lists of Network (IP) Addresses. These lists are used to specify [Client](#) and [Blacklisted](#) addresses, to specify access restrictions for [Listeners](#), etc.

This section describes the format used to specify Network Address Lists.



A Network Address List is specified as multi-line text data.

Each line should include either:

- 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 ! sign. In this case the specified address or the address range is excluded from the list composed using the preceding lines.

A comment (separated with the semicolon (;) symbol) can be placed at the end of a line.

Lines starting with the semicolon symbol and empty lines are comment lines.



Listeners

The CommuniGate Pro Server accepts SMTP, IMAP, POP, LDAP, and other TCP/IP connections using Listeners. A Listener opens one or several listener sockets, each socket accepting TCP/IP connections directed to the specified port number and, optionally, to the specified local IP address.

The Listener settings allow the Server administrator to specify the type of connections to accept (regular, clear text connections or secure connections), and the optional remote address restrictions - that grant access to the listener sockets only to the computers inside the specified networks.

CommuniGate Pro TCP Listeners can limit the number of incoming connections that come from the same IP address. This can help to prevent some of the Denial of Service (DoS) attacks.

CommuniGate Pro Server can accept SNMP, RADIUS, SIP, and other UDP connection-less requests using UDP Listeners. A UDP Listener opens one UDP listener socket and receives UDP packets on the specified port number and, optionally, on the specified local IP address. The UDP Listeners support address restrictions.

Multi-Socket Listening

TCP/IP services use Listeners to accept incoming connections. Each Listener can use one or several *listener sockets*. A listener socket accepts incoming TCP connections on the TCP port number you specify. You should also specify if the socket should accept connections 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.yy.zz.tt, then you can create a different listener socket on the same TCP port N and a different local address xx.yy.zz.tt.

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 Address	Init SSL/TLS	Remote Address Restrictions
143	all addresses ▼	off ▼	Grant ▼ 10.0.0.81-10.0.0.90; internal LAN 10.0.1.12 ; gateway
993	all addresses ▼	on ▼	None ▼
0	all addresses ▼	off ▼	None ▼

To create a new listener socket, change the value in the last table element from 0 to the desired TCP 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 services: one for regular, *clear text* connections and one (on a different port number) for secure connections (see below).

Secure Sockets

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 came encrypted: [clear-text Login](#) operations are considered secure, STARTTLS operations are prohibited, etc.

Restrictions

You may want a listener socket to accept connections only from the certain remote network (IP) addresses.

If you set the Restriction setting to `Grant`, and list the IP addresses in the text field, the socket will accept connections 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 [System Administrator](#) 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.

Limiting Connections from the same Address

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:

Limit Connections from same Address:	<input type="text" value="10"/>	Reserve Connections for Clients:	<input type="text" value="10"/>
---	---------------------------------	---	---------------------------------

When you set this 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 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.

UDP Listeners

UDP services use UDP Listeners to accept incoming packets (connection-less requests). Each UDP Listener uses a *UDP listener socket*. A UDP listener socket accepts incoming UDP packets on the UDP port number you specify. You should also specify if the socket should accept packets on all IP addresses your server computer has, or

only on a selected address.

Because of the nature of UDP sockets, you cannot have two UDP listener sockets that use the same UDP port number and the same local IP address: if you create a listener socket on the UDP 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 UDP port N and a specific local address xx.yy.zz.tt, then you can create a different listener socket on the same UDP port N and a different local address xx.yy.zz.tt.

The UDP Listener WebAdmin pages allow you to configure UDP Listeners:

Port	Local Address		Remote Address Restrictions
161	all addresses	Grant	10.0.0.81-10.0.0.90; internal LAN 10.0.1.12-10.0.1.13 ; NAS boxes

All UDP Listener settings have the same meaning as the TCP Listener settings (see above).



Dialup

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 mail:

- 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 a Web browser to open the TCP Activity Schedule page. Open the Network pages in the Settings realm of the WebAdmin Interface, and then follow the Schedule link.

Log: Major & 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 settings 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:

- 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`.



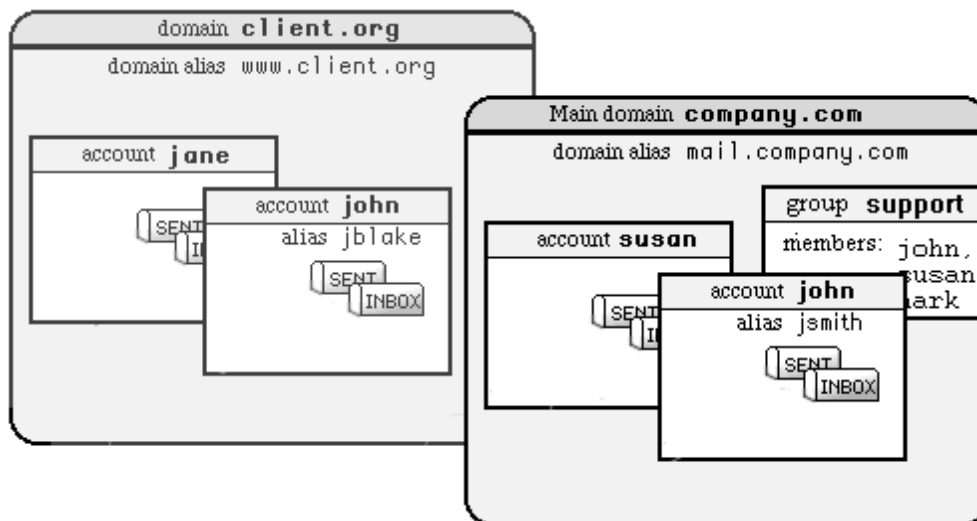
Objects

CommuniGate Pro has a hierarchy of *objects* it serves on each installation. On the topmost level there is a set of Domains, and each Domain contains accounts, groups, mailing lists and forwarders. Each account contains one or several mailboxes. Each mailbox contains some number of E-mail messages.

Besides these basic objects, CommuniGate Pro supports supplementary objects: Accounts can contain File Sites and Preference Data, Domains can contain certificates and WebUser Skin files, etc.

Domains

Domains are the CommuniGate Pro objects that contain other objects: accounts, mailing lists, groups and forwarders. 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, that 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. 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.

To browse the list of all Main Domain Objects, open the Accounts section of the [WebAdmin Interface](#).

If you want to view Objects in a [secondary Domain](#), open the Domains section, and follow the link for that Domain. You should have the Can Modify All Domains And Accounts [access right](#) to browse, create, and remove Accounts and to modify Account settings.

If you are a [Domain Administrator](#), then the list of Objects in your Domain appears on the main [Domain Administration](#) page.

<div>Display</div>		<div>1000</div>	<div>Filter: smith</div>	
<div><input checked="" type="checkbox"/></div>	Accounts (2 of 345)	<div><input checked="" type="checkbox"/></div>	Account Details	<div><input checked="" type="checkbox"/></div>
		<div><input checked="" type="checkbox"/></div>	Groups (1 of 4)	<div><input checked="" type="checkbox"/></div>
			Forwarders (1 of 10)	<div><input type="checkbox"/></div>
				Aliases (0 or 5)
Object	Type	Storage	Last Access	Devices
lsmith	MultiMailbox	140K	19-May-06 [10.0.8.198]	2
smith-L	Group		15	
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 options allow you to specify the type of Objects you want to display: Account, Groups, Forwarders, Aliases, and show 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 sotrage 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 Group, the group members number is displayed.
- If the object is a Forwarder, the forwarding address is displayed.

Accounts

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 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.

Accounts are created inside CommuniGate Pro *domains*.

Each CommuniGate Pro domain has its own set of accounts. Accounts should have unique names within their domain, but two accounts in different domains can have the same name.

Account E-mail address is `accountname@domainname` address 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 this account address are delivered to the account using the [Local Delivery](#) module.

An Account may have several names (for example, `john.smith` and `jsmith`). An administrator can create [account aliases](#) to assign several names to one Account.

Each CommuniGate Pro Account has its own settings and a set of [Mailboxes](#).

Each CommuniGate Pro Account has its own [Personal File Site](#).

Accounts can also store additional information and data. See the [Account Data](#) section for the details.

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.

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.

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 pro-

cessing, archiving and digesting, browsing, posting policies, moderating, etc.

See the [LIST](#) section for more information about CommuniGate Pro Mailing Lists.

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, 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.

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.



Domains

CommuniGate Pro Server can serve Accounts in its Main Domain, and, optionally, in multiple Secondary Domains, each with its own set of user Accounts (and other objects such as Mailing Lists, Groups, and Forwarders).

Every domain can have one or several *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 mail 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 mail domains 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 mail domain.

See the [Mapping](#) section for the details.

When a client application (a mailer) connects to your CommuniGate Pro Server, and specifies an account name, the Server has to detect in which Domain to look for that Account.

You can use multiple domains if:

- your Server has multiple IP addresses (i.e. it uses *multihoming*), and each Domain has its own (dedicated) IP address on your Server computer. In this case the Server will detect the IP address to which a client connects and look for accounts in the Domain associated with that IP address.

or

- your clients use the [Web User Interface](#)

or

- your clients use mailers that can specify the full account name (i.e. an account name and the domain name) when connecting to a server.

See the [Access section](#) for the details.

Displaying the Domain List

To display the list of all Domains served with your server, use a Web browser and enter the Domains section. You should be connected as the Postmaster or any other user with the `Can Modify All Accounts` and `Domains` access rights.

Display	30	Filter:					
2083 Accounts	3 of 3 Domains selected	<input checked="" type="checkbox"/>	Show Aliases	3 selected			
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 domain accounts have been opened (since the Server last restart), and the last time any domain account was opened.

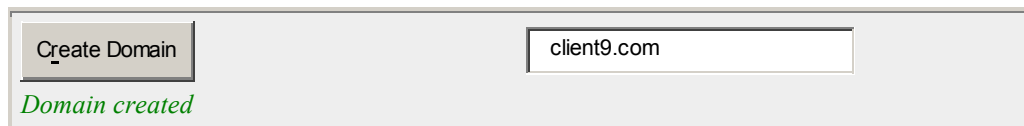
Select the `Show Aliases` option to include domain aliases into the list. Each domain alias list element contains the link to its "real" domain account list and settings pages.

Click a domain name to view the [accounts](#) in that domain.

Click the word Settings in the last column to view and update the domain Settings.

Creating a New Domain

Type a new domain name into the field on the right side of the Create Domain button.



A screenshot of a web interface showing a button labeled "Create Domain" and a text input field containing "client9.com". Below the button, the text "Domain created" is displayed in green.

Click the Create Domain button. When a new domain is created, its name appears in the Domain List.

If a server is a member of a [Dynamic Cluster](#), the additional Create Shared Domain button appears. Click that button to create a domain that will be served by all members of the Cluster. The domain created using the Create Domain button are created as "local" domains and are served by this server only.

Specifying Domain Settings

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 Accounts List page.



A screenshot of a web interface showing two dropdown menus. The first is labeled "Account Log:" and the second is labeled "Mailbox Log:". Both dropdown menus are set to "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 Accounts link to switch to the domain Account List.

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 and IMAP mailer 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	
<div> <div>All Available</div> <div>▼</div> </div>	[206.40.74.198]

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.

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 mailer settings. See the [Access](#) section for the details.

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.

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 mailer 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.

Enabling Services

Each Domain has a set of settings that specify which CommuniGate Pro services can be used with the Domain Accounts:

Enabled Services																			
<input checked="" type="checkbox"/>	Mail	<input checked="" type="checkbox"/>	POP	<input checked="" type="checkbox"/>	IMAP	<input type="checkbox"/>	PWD	<input checked="" type="checkbox"/>	ACAP	<input checked="" type="checkbox"/>	WebMail	<input checked="" type="checkbox"/>	WebSite	<input checked="" type="checkbox"/>	Relay	<input checked="" type="checkbox"/>	Mobile	<input checked="" type="checkbox"/>	FTP
<input checked="" type="checkbox"/>	MAPI	<input checked="" type="checkbox"/>	TLS	<input type="checkbox"/>	SMIME	<input checked="" type="checkbox"/>	LDAP	<input checked="" type="checkbox"/>	WebCal	<input type="checkbox"/>	RADIUS	<input checked="" type="checkbox"/>	SIP	<input checked="" type="checkbox"/>	PBX	<input checked="" type="checkbox"/>	XMPP	<input checked="" type="checkbox"/>	XIMSS

Mail

If this option is disabled, incoming mail is not delivered to Domain Accounts. 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.

POP, IMAP, PWD, ACAP , XMPP, XIMSS

If a protocol option is disabled, Accounts in this Domain cannot be opened using that protocol.

WebMail

If this option is disabled, the Domain Accounts cannot be opened using the WebUser Interface, and the Domain mailing lists cannot be browsed.

WebSite

If this option is disabled, the Personal File Sites in this Domain cannot be accessed via HTTP.

FTP

If this option is disabled, FTP access to this Domain is disabled.

MAPI

If this option is disabled, MAPI access to this Domain is disabled.

Mobile

If this option is disabled, Domain users will not be able to connect to their Accounts from Internet Addresses not included into the [Client Addresses](#) list. This can be useful if you provide free Accounts in this Domain and you want Domain users to connect to those Accounts only from the dial-up addresses your own site provides.

Relay

If this option is disabled, Domain users will not be able to use the [Mobile Users Support](#) features. This can be useful if you provide free WebMail Accounts in this Domain and you do not want spammers to use these Accounts to enable SMTP relaying.

TLS

If this option is disabled, secure (SSL/TLS) access to Accounts in this Domain is disabled.

SMIME

If this option is disabled, Secure Mail (S/MIME) features implemented in the WebUser Interface are not available for Accounts in this Domain.

LDAP

If this option is disabled, users of this Domain cannot authenticate themselves with the [LDAP](#) module ("BIND") using their account names.

WebCal

If this option is disabled, users of this Domain cannot use the [Calendaring](#) functions of the WebUser Interface.

RADIUS

If this option is disabled, users of this Domain cannot be authenticated using the [RADIUS](#) module.

SIP

If this option is disabled, users of this Domain cannot use the [SIP](#) operations that require authentication.

PBX

If this option is disabled, users of this Domain cannot use the [PBX](#) services.

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:	<input type="text" value="10000"/> ▼	390
Mailing Lists:	<input type="text" value="5"/> ▼	5
RPOP Accounts:	<input type="text" value="25"/> ▼	15
MAPI Connections:	<input type="text" value="50"/> ▼	37

Domain Aliases

Each CommuniGate Pro domain can have aliases (alternative names). If the domain `client.dom` has the `mail.client.dom` and `www.client.dom` aliases, mail 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, and other mailer applications the account names `user@mail.client.dom` and `user@www.client.dom` can be specified in the mailer 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 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` alias for the `client.dom` domain.

Aliases
<input type="text" value="www.client.dom"/>
<input type="text"/>

You can modify existing 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 list of domain aliases.

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.



The screenshot shows a web interface panel titled "Directory Integration". Inside the panel, there is a "Usage:" label followed by a dropdown menu currently set to "Keep In Sync". Below this, there are two buttons: "Delete All" and "Insert All".

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.

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 Authenticator:	Yes	
Mail to Unknown Names is:	Rerouted to	postmaster@client1.com
Signals to Unknown Names are:	Rejected	

Consult External Authenticator

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 Names

This setting specifies what the Server should do when unknown account/object names are encountered in 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.

Signals to Unknown Names

This setting specifies what the Server should do when unknown account/object names are encountered in Signal (SIP) addresses. This setting is set in the same way as the Mail to Unknown Names 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.

Mail to <all@client1.com>	
is distributed for	Authenticated Users
Send to Forwarders:	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 *trusted* methods (Web User Interface, 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 **Web User Interface** settings:

WebUser Interface	
Mail Trailer Text:	<div>-+-+-+----- Get an E-mail account at http://mail.mycompany.com</div>
WebSite Prefix:	<div>~</div>
Web Banner Text:	<div><CENTER> <IMG SRC="http://w w w .stalker.com/banner.gif"Border=0: </CENTER></div>

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 for Personal File Sites. It can be also changed to an empty string, so URLs for Personal File Sites will look like `http://domainName:port/accountName/`. See the [HTTP](#) module description for more details.

Web Banner Text

The text in this field is inserted (optionally) into the beginning of all HTML files retrieved from the Domain user [Personal File Sites](#).

Enabling Auto-Signup

You can allow users to create domain accounts themselves, via the [WebUser](#) Interface:

Auto Sign-up	
<input checked="" type="checkbox"/>	Enabled

If the Auto Sign-up option is enabled, the Sign-up link appears on the domain Login Web page. This link allows new users to open the Sign-up page, where they can enter a new account name, the user real name, and the desired password.

The Server checks that an account with the specified name does not exist 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.

SMTP Options

The SMTP panel controls how E-mail messages are sent from and received for Accounts in this Domain.

SMTP	
Send via:	default (any) <input type="button" value="v"/> IP Address
Force AUTH for:	non-clients <input type="button" value="v"/> when Receiving
Account Check:	default (Disabled) <input type="button" value="v"/>

Send via

Use this setting tell the [SMTP](#) module to use a particular Local IP Address to send mail originated from this Domain. For example, if there is a risk that some of the Domain users can be involved in spamming activity, you may want to send all the Domain mail via an IP address dedicated to that Domain, so the other IP Addresses of your Server will not be blacklisted.

The available options include:

any

The SMTP module should not use any particular IP Address when sending messages originating from this Domain. The module allows the Server OS to select some Server Local IP Address for outgoing SMTP connections.

first

For all messages received for or generated in this Domain, the SMTP module should use the first IP Address from the set of IP Addresses assigned to this Domain.

Note: in a Cluster setup, the Server selects the first assigned IP address that is local for the Cluster member actually sending the message.

same

The SMTP module should try to use the same Local IP Address that was used to receive the message. If the message was received using the SMTP protocol or using the XTND XMIT extension of the POP3 protocol via a Server Local IP Address assigned to this Domain, the same Server Local IP Address will be used to send the message out (to relay it).

In all other cases, the first IP address assigned to the Domain (see above) will be used.

Note: this option should NOT be used if you have a firewall and some of the Server Local IP Addresses belong to an internal LAN: users can submit messages via internal LAN IP addresses, and the SMTP module will fail to send those messages to the Internet if it has to use an internal LAN IP address.

Note: in most cases, the First option (see above) provides better results.

Force AUTH

If this option is enabled and the SMTP module receives a message 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.

Account Check

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.



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 [external INBOX](#) mailboxes. These mailboxes are stored not inside the CommuniGate *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 a domain, you should specify the External INBOX settings:

Server OS Integration	
External INBOX	<div>location:  default ()</div> <div> <input type="text" value="/var/mail/dj/*"/></div>

location

This setting specifies where the external INBOX files should be located. For each Account that has an external INBOX, the Server substitutes the asterisk sign (*) with the CommuniGate 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.

synchronize using

This setting specifies the file locking method to use for updates synchronization.

See the [Sharing](#) 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 have Kerberos keys enabling secure single signon 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.

Domain Rules

Domains can have Automated Rules that are applied to all messages 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	
<input type="text" value="other.domain.com"/>	

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 with a Web browser, and enter a new account name into the New Domain Name field. 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 with a Web browser, and click the Remove Domain button. The confirmation page should appear. 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, and all its accounts will be permanently removed from the Server disks.

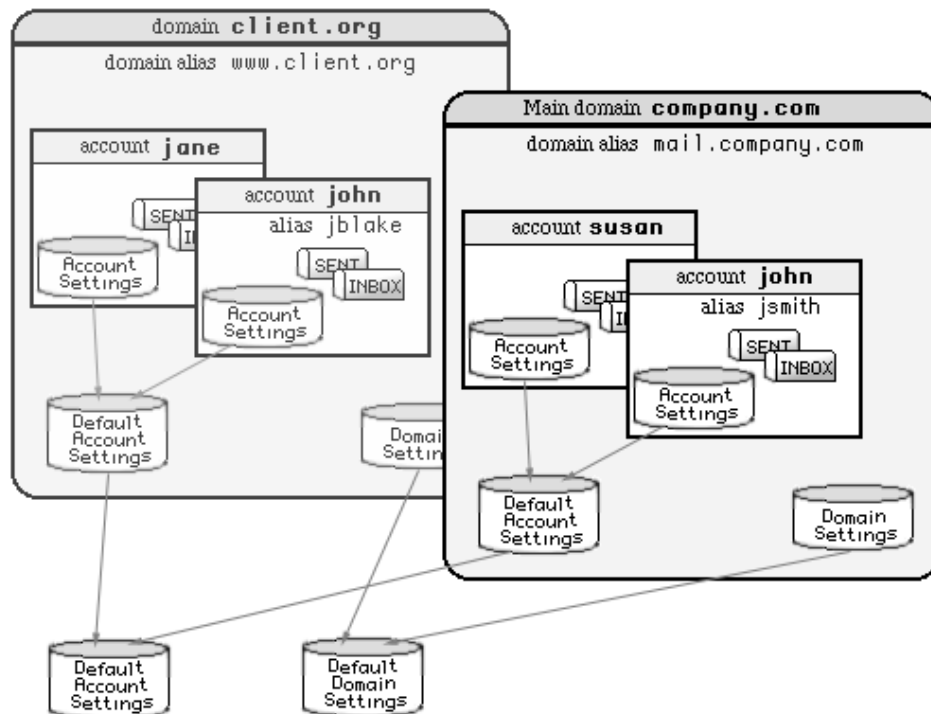
You cannot remove a domain when any of its accounts is in use.

Specifying Default Domain Settings

A domain setting can have the `default` value. In this case the actual setting value is taken from the global 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.



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 are stored as subdirectories of the `SharedDomains` directory.

If your server or cluster serves many (more than 3,000) Domains, you may want to create additional [Domain Subdirectories](#) inside the Domains and/or 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:	flat ▼
Rename In Place:	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.

Note: if you cannot store all Domains on one disk volume, you can copy some `xx.sub` directories to other volumes, and replace them with symbolic links.

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 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) ▼

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.

Note: many other mail systems process large Domains with account subdirectories, too. Every time an account is to be opened, those systems form the account subdirectory name using some built-in method. As a result, the built-in method cannot be changed "on the fly", and accounts cannot be moved between subdirectories. The CommuniGate Pro Server uses its subdirectory name forming methods only when a

new Account is being created or when an Account is being renamed, and it always remembers in which subdirectory every Account is located. The Server does not have to form the subdirectory name every time an Account is to be opened. As a result, the CommuniGate Pro domain "foldering" methods can be changed at any moment, and the Accounts can be moved between the subdirectories when the server is not running.

Note: if you cannot store all Domain Accounts on one disk volume, you can copy some `xx.sub` directories to other volumes, and replace them with symbolic links.

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.



Mapping

Many domain names can have DNS records pointing to your Server. But the Server automatically processes mail sent only to its Main Domain, or to one of its [Secondary Domains](#).

To let the Server accept and locally process mail sent to any other domain, that domain should be *mapped* to any existing CommuniGate Pro domain. Otherwise, a message sent to an unlisted domain will be directed to the SMTP module for relaying, and that module, detecting that it has to send the message to itself, will reject the message reporting about a 'DNS loop'.

CommuniGate Pro provides a variety of methods to serve multiple domains. The Server Administrator should decide how to serve each domain and either create a new full-scale [Secondary Domains](#) or just map a mail domain name onto one of the existing CommuniGate Pro Domains.

CommuniGate Pro Domains

The CommuniGate Pro Server allows you to create [Secondary Domains](#) in addition to the Main Server Domain. In this section the Main and Secondary domains are referenced as *real* or *CommuniGate Pro* domains.

Each CommuniGate Pro domain has its own set of accounts, own settings, own WebUser Interface, etc. If `client1.com` and `client2.com` are CommuniGate Pro domains, both domains can have the account `info`, and these Accounts are different - each one with its own settings, mailboxes, passwords, etc.

Each CommuniGate Pro Domain can have its own [Domain Administrator\(s\)](#).

Direct Mapping (Domain Aliases)

Very often one real Domain should have several aliases (additional names). For example, if your CommuniGate Pro has the `company.com` Domain, you may want the to process the `mail.company.com` and `relay.company.com` domain names as aliases for the `company.com` Domain.

To create a Domain Alias, open the `company.com` [Domain Settings](#) page and enter the `mail.domain.com` name into the [Aliases](#) table. Click the Update button to see the alias name accepted, and enter the `relay.company.com` name into the new empty field. Click the Update button again.

Created aliases will tell the [Router](#) that all references to these two domain names should be substituted with the `company.com` name. As a result, messages sent to `info@company.com`, `info@mail.company.com`, and to `info@relay.company.com` will all end up in the `info` Account in the `company.com` CommuniGate Pro Domain.

The Router and Domain Aliases are also used for [account access](#), so when a user tries to access the `info@mail.company.com` account, the `info` Account in the `company.com` CommuniGate Pro Domain is opened.

You can use the [Router domain records](#) to achieve the same results:

```
mail.company.com = company.com
relay.company.com = company.com
```

You can also use the [Router alias records](#):

```
<*@mail.company.com> = *@company.com
<*@relay.company.com> = *@company.com
```

Modifying Mapping

Sometimes you do not want to create a separate CommuniGate Pro Domain for a mail domain (for example, if that mail domain will have only few accounts), but you do not want the account names in that mail domain to interfere with the account names in the CommuniGate Pro Domain you map it on.

For example, your client with `client.com` CommuniGate Pro Domain wants to accept mail for the

`info@shop.client.com` and `order@shop.client.com` addresses, but these accounts should not be the same as the `info` and `order` Accounts in the `client.com` Domain.

Use the Router *alias record* to map the `shop.client.com` mail domain:

```
<*@shop.client.com> = shop-*@client.com
```

This Router record will redirect mail sent to `info@shop.client.com` to `shop-info@client.com`. Mail sent to `order@shop.client.com` will be redirected to `shop-order@client.com`, etc.

The Accounts `shop-info` and `shop-order` must exist in the `client.com` Domain.

To access these accounts, the users should specify the `client.com` name as the name of their mail server, and `shop-info`, `shop-order`, etc. as their account names.

You can use this method to modify just some of the additional domain account names, while mapping other names directly to the names in the `client.com` CommuniGate Pro domain:

```
<info@shop.client.com> = shop-info@client.com
```

```
<order@shop.client.com> = shop-order@client.com
```

```
<*@shop.client.com> = *@client.com
```

Please note that the generic alias record must be specified *after* the first two records.

Unified Domain-Wide Accounts

You can tell the CommuniGate Pro Server to store all mail for a mail domain in one Account. This method is useful if:

- the mail 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 mail domain has only few POP3 users and you do not want to create a full-featured CommuniGate Pro domain to serve them.

To create a Unified Domain-Wide Account for the `client.com` domain, use the following Router *domain record*:

```
client.com = client.local
```

or a Router *alias record*:

```
<*@client.com> = *@client.local
```

All messages to the `client.com` mail domain will be stored in the `client` Account in the CommuniGate Pro

Main Domain. The `.local` suffix explicitly tells the [Local Delivery Module](#) to accept this address and direct it to the `client` account.

The Local Delivery Module uses the *local part* of the address to form the `X-Real-To:` header fields in the stored messages. These fields will allow the client software to retrieve messages from the CommuniGate Pro Unified Domain-Wide Account and distribute them locally to the proper users of that mail domain. See the [Local Delivery Module](#) section for more details.

If users of such mail domain do not have their own mail server that retrieves mail from the Unified Domain-Wide Account, but connect to your CommuniGate Pro Server directly, the [POP module](#) will show each user only the messages directed to that user, rather than all messages stored in the Unified Domain-Wide Account. See the [POP Module](#) section for more details.

Please note that while the following Router record will also store all mail sent to the `client.com` domain in one `client` account:

```
<*@client.com> = client
```

But this Router record discards the information about the original user name (the part before the @ sign). As a result, no `X-Real-To:` header fields will be added to the messages stored in the `client` account.

You can mix the mapping methods with the Unified Domain-Wide Account method. For example, if you want messages sent to the `jim@client.com` address to be stored in the `client-jim` account in the Main CommuniGate Pro domain, while directing the rest of the `client.com` mail to the Unified Domain-Wide Account `client`, use the following Router records:

```
<jim@client.com> = client-jim  
<*@client.com> = *@client.local
```

If you are serving many small mail domains, providing Unified Domain-Wide Accounts can be a very effective alternative to real CommuniGate Pro domains.



Accounts

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 Server Domain. The Master (unlimited) access right is granted to that Account.

The `pbx` Account is automatically created in the Main Server Domain. See [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.

The screenshot shows a web form for creating an account. At the top left is a button labeled 'Create Account'. To its right is a text input field containing the username 'jsmith'. Below the button is a dropdown menu labeled 'MultiMailbox'. To the right of the dropdown is a checkbox, which is currently unchecked, with the label 'external INBOX in /var/mail/*' next to it.

Use the pop-up menu to specify the Account type:

MultiMailbox

A folder-type Account that can contain several mailboxes of various types. The INBOX mailbox is automatically created within the new Account. All incoming mail is stored in the INBOX mailbox by default. The user can create additional mailboxes using any IMAP client software, or using the CommuniGate Pro Web E-mail Interface.

Text Mailbox, MailDir Mailbox, ...

An Account that contains a single INBOX mailbox. 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 signs - some mail systems cannot send mail to E-mail addresses containing other symbols.

external INBOX

Select this option if you want the new Account INBOX to be created as an [external mailbox](#), so new Account can be used with legacy *local mailers*. This option is enabled only if the external mailbox location is specified in the Domain Settings.

Click the Create Account button. When a new Account is created, its name appears in the Accounts 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.

Specifying Account Settings

To specify Account Settings, click the Account name link in the Accounts list. The Account Settings page appears.

Real Name:	<input type="text" value="John F. Smith"/>
organization:	<input type="text" value="ACME Industries, Inc."/>
city:	<input type="text" value="Mill Valley"/>
CommuniGate Password:	<input type="password" value="*****"/>
telephoneNumber:	<input type="text" value="800 262 4722"/>

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 Allow to Use: Yes ▼ Allow to Modify: Yes ▼ Encryption: A-crpt ▼	Server OS Integration OS UserName:  default (*)  <input type="text"/> Enable OS Password: default (No) ▼
Kerberos: default (Enabled) ▼ Certificate: default (Disabled) ▼	External Authentication Allow to Use: default (No) ▼

CommuniGate Password

Allow to Use

This setting tells the Server if it should use the CommuniGate Password string when authenticating a user. The user may use the CommuniGate Password, or the Server OS password (see below) to connect to the CommuniGate Pro Server.

Allow to Modify

This option allows the user to modify the CommuniGate Password via either the [PWD module](#) or via the [WebUser Interface](#) for Account Settings.

Encryption

This option 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.

The `U-crpt` password encryption is available on Unix platforms only. It is used for compatibility with the Unix "crypt" encryption method and it should be used for migrating users from other mail servers only. The U-crtp-encrypted passwords can not be used for Secure (SASL) Authentication methods.

See the [Security section](#) for the details.

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).

Server OS user name

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 sign, 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 `jmsmith`, 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`.

Enable OS Password

This setting allows the user to work with the Account using the password set in the Server OS registration information for this user. If both OS and CommuniGate Pro passwords are enabled, and if at least one of those passwords matches the password provided by a user, the user is allowed to connect to the Account.

See the [Security](#) section for the details.

Kerberos

This setting allows the user to log into the Account using the Kerberos Authentication method.

See the [Security](#) section for the details.

Certificate

This setting allows the user to log into the Account using the Client Certificate Authentication method.

See the [PKI](#) section for the details.

External Authentication: Allow to Use

This setting allows the user to work with the Account using the 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 mailer application connects to the Server and supplies a password for this account using an unsecure ("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.

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.

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 of 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 global Default Account Settings.

Enabled Services

There is a set of settings that specify which CommuniGate Pro services can be used with the Account:

Enabled Services																				
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Mail	<input checked="" type="checkbox"/>	POP	<input checked="" type="checkbox"/>	IMAP	<input checked="" type="checkbox"/>	PWD	<input type="checkbox"/>	ACAP	<input checked="" type="checkbox"/>	WebMail	<input checked="" type="checkbox"/>	WebSite	<input checked="" type="checkbox"/>	Relay	<input checked="" type="checkbox"/>	Mobile	<input checked="" type="checkbox"/>	FTP
<input type="checkbox"/>	<input type="checkbox"/>	MAPI	<input checked="" type="checkbox"/>	TLS	<input checked="" type="checkbox"/>	SMIME	<input type="checkbox"/>	LDAP	<input checked="" type="checkbox"/>	WebCal	<input type="checkbox"/>	RADIUS	<input checked="" type="checkbox"/>	SIP	<input checked="" type="checkbox"/>	PBX	<input checked="" type="checkbox"/>	XMPP	<input checked="" type="checkbox"/>	XIMSS

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 global Default Account Settings.

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 in the Domain Settings, no account in that domain can be accessed via these protocols.

Mail Settings

Mail		Used
Mail Storage:	<input type="text" value="default (30 Mbytes)"/>	134K
Mailboxes:	<input type="text" value="default (Unlimited)"/>	33
New Mailbox Format:	<input type="text" value="default (TextMailbox)"/>	

Mail Storage

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.

Mailboxes

This option is used to specify the maximum number of mailboxes that can be created in this Account.

Mailboxes

This option is used to specify the maximum number of Mailboxes that can be created in this Account.

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.

Mail	
Delay New Mail if:	<input type="text" value="default (100)"/> % full
Send Alerts if:	<input type="text" value="80"/> % full
Send Notice if:	<input type="text" value="90"/> % full

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.

	Mail	Used
Allowed Mail Rules:	Filter Only ▼	7
RPOP modifications:	No ▼	3
Accept Mail to all:	default (Yes) ▼	
Add Mail Trailer:	default (Yes) ▼	

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, and Store 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.

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](#).

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 global Default Account Settings.

Signal Settings

	Calls	Used
Allowed Call Rules:	Filter Only	7
Registered Devices:	default (10)	3

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.

Registered Devices

This setting specifies the maximum number of "Contacts" (devices) the Server can register for this Account.

File Storage Settings

	Files	Used
File Storage:	<input type="text" value="1 Mbyte"/>	89K
Files:	<input type="text" value="default (25)"/>	6
Add Web Banner:	<input type="text" value="default (Yes)"/>	
Default Web Page:	<div><div></div><div>default (default.html)</div></div> <div><div></div><div><input type="text" value="index.html"/></div></div>	

File Storage

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.

Files

This option is used to specify the maximum number of all files in the Account [File Storage](#).

Add Web Banner

This setting tells the Server to insert the Web banner code (specified in the [Domain Settings](#)) to all HTML files retrieved from the 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.

Account Aliases

Each Account can have Aliases (alternative names). If the Account JohnSmith has the jsmith and j.smith Aliases, mail directed to jsmith and to j.smith will be stored in the JohnSmith Account. Also, to access the JohnSmith Account via POP, IMAP, and other mailer application the user names jsmith and j.smith can be specified in the mailer settings.

Aliases
j.smith
jsmith

You can modify existing 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 list of Account aliases.

Alias names should not be the same as the name of some other Account, or other Object in the same Domain.

You can specify several Aliases in one field, by separating them with the comma signs.

Account Telephone Numbers

Each Account can have one or server Telephone (PSTN) numbers assigned to it.

The Server maintains a global list of all Telephone Numbers assigned to all its 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.

See the [PSTN](#) section for more details.

Creating Mailing Lists

Every CommuniGate Pro [Mailing List](#) has an owner - an Account in the main or one of the secondary Domains. To create a Mailing List, you should create the Owner Account first. For each list, the Mailing List manager creates several mailboxes inside the owner Account, so the owner Account should be of the MultiMailbox type.

Mailing Lists		Create List	<input type="text"/>
List Name		Subscribers	
RD-List		1025	

To create a mailing list, type the list name and click the Create List button. To modify the list settings, to rename and remove the mailing lists use the links to the [Mailing List Settings](#) pages.

Renaming Accounts

If you want to rename an Account, open its Settings page with a Web browser, 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.

<input type="button" value="Remove Account"/>	New Account Name:
	<input type="text"/>
	<input type="button" value="Rename Account"/>

Removing Accounts

If you want to remove an Account, open its Settings page with a Web browser, 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.

Default Account Settings

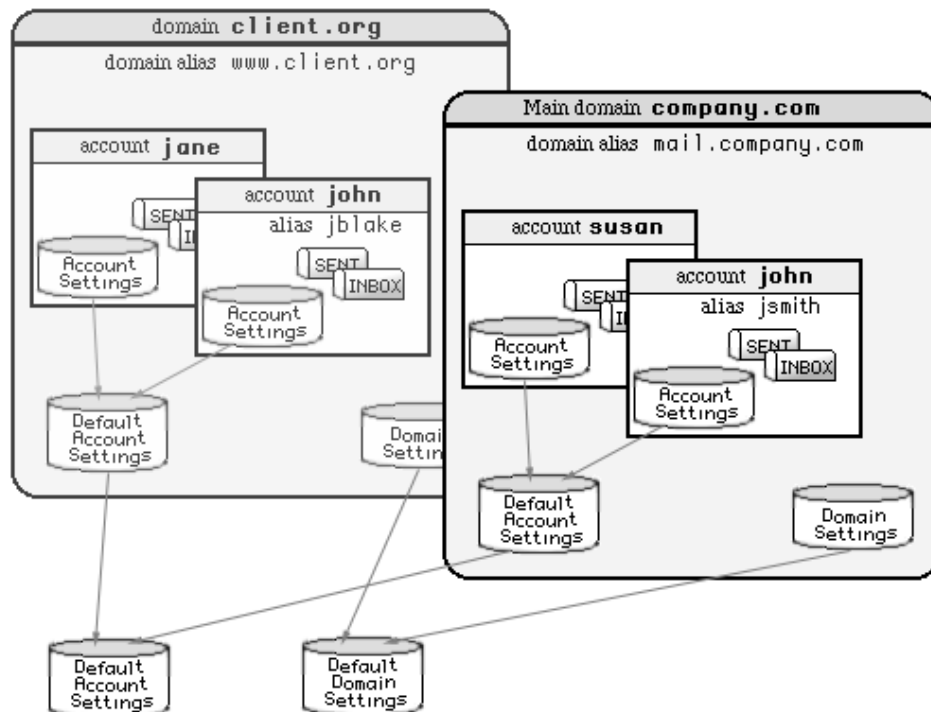
An Account setting can have the `default` value. In this case the actual setting value is taken from the Default Account Settings for this domain. You can modify these default values by clicking the Account Defaults link on the Account List or Domain Settings page.

The Default Account Settings page resembles a regular Account Settings page. Any setting on that page can also be set to the `default` value, in this case the actual value is taken from the server-wide Default Account Settings, which specify the default setting values for all Accounts in all Server Domains.

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.

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		Lock
<input type="text" value="Sent"/>	<input type="text"/>	<input type="checkbox"/>
<input type="text" value="Drafts"/>	<input type="text"/>	<input type="checkbox"/>
<input type="text" value="Calendar"/>	<input type="text" value="IPF.Appointment"/>	<input checked="" type="checkbox"/>
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>

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	
<input type="text" value="INBOX"/>	
<input type="text" value="Sent"/>	
<input type="text" value="~public/new s"/>	
<input type="text"/>	

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	Foreign Mailbox Name
<input type="text" value="mkt-new s"/>	<input type="text" value="~marketing/announcements"/>
<input type="text"/>	<input type="text"/>

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.

Initial Greeting Message	
<div><div>From: postmaster@mail.stalker.com (Server Postmaster)</div><div>To: New Subscriber</div><div>Subject: Welcome to our wonderful server!</div><div>MIME-Version: 1.0</div><div>Content-Type: text/plain</div><div>Dear Customer,</div><div>We are happy to see you with us! If you ever have any problem with our service, please send e-mail to:</div></div> <div><div></div><div></div><div></div><div></div></div>	

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 `Date :` header field is automatically added to the stored messages.

If the message contains non-ASCII symbols, then the message text should start with `[charsetName]` string, and the Content-Type header field should include the `charset=` parameter:

Initial Greeting Message

[ISO-8859-1]
From: postmaster@mail.stalker.com (Server Postmaster)
To: New Subscriber
Subject: Bienvenue à notre serveur merveilleux!
MIME-Version: 1.0
Content-Type: text/plain; charset=ISO-8859-1

Cher Client,

Nous sommes heureux de vous voir avec nous!

Templates can be used to generate an initial Personal File Site page for all newly created Accounts:

Initial Personal Home Page

```
<HTML>
<HEAD></HEAD>
<BODY>
<H1>This is my personal Web Page.</H1>
</BODY>
</HTML>
```

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 Personal Web File area of every 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.

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 4.5 and later Macintosh 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-crypt` encrypted passwords. See the Migration section for more details.

Storage

This column contains the maximum Account 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; several Aliases may be specified in one field if they are separated with the comma signs.

Rules

This column contains the Account [Message 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.

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 Site 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 Personal File Site, 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.



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.

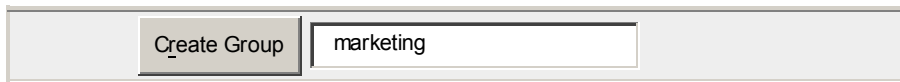
Real-time (IM, audio, video, etc.) requests sent to a Group are processed with the [Signal](#) component. These requests are "forked" to all Group members.

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.

A screenshot of a user interface element for creating a group. It consists of a light gray rectangular container. Inside, on the left, is a button with a dark gray border and the text "Create Group". To the right of the button is a white text input field with a dark gray border, containing the text "marketing".

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 in the Accounts list. The Group Settings page appears.

Real Name: <input type="text" value="R&D Group"/>	
<input type="checkbox"/> Report Delivery to Group	<input checked="" type="checkbox"/> Set Reply-To to Group
<input checked="" type="checkbox"/> Expand Member Groups	<input checked="" type="checkbox"/> Reject Automatic Messages
<input checked="" type="checkbox"/> Remove Author from Distribution	<input checked="" type="checkbox"/> Remove To and Cc from Distribution
<input checked="" type="checkbox"/> Disable E-mails	<input checked="" type="checkbox"/> Disable Signals

Members	
<input type="text" value="john"/>	
<input type="text" value="jim"/>	
<input type="text" value="susan"/>	
<input type="text" value="bill@partner.dom"/>	
<input type="text"/>	

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 sign. 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 or a Group).

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.

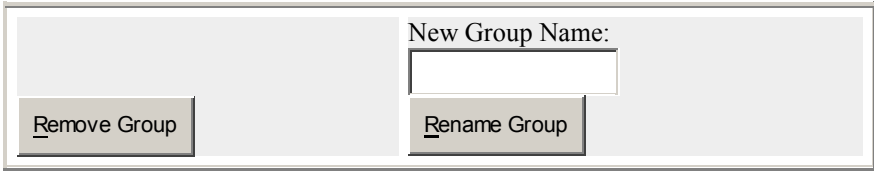
When an Account user is trying to open a [Foreign Mailbox](#), the Server checks the [Mailbox Access Rights](#) granted to all Groups containing that Account.

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 with a Web browser, and enter a new Group name into the New Group Name field. 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.



The screenshot shows a web interface for group settings. It features a light gray background with a darker gray border. On the left, there is a button labeled "Remove Group". On the right, there is a text input field labeled "New Group Name:" above it. Below the input field is another button labeled "Rename Group".

Removing Groups

If you want to remove a Group, open its Settings page with a Web browser, and click the Remove Group button.



Forwarders

CommuniGate Pro domains can contain forwarder objects. A Forwarder is just some E-mail address associated with the Forwarder name.

Forwarders work exactly as the [Router](#) alias records: a Forwarder "jim" in the domain client1.com that contains the E-mail address "john@otherdomain.dom" acts as the 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:

Create Forwarder	marketing	Forward to:	mark-dept@contractor.example
------------------	-----------	-------------	------------------------------

To create a new Forwarder enter its name and the address to forward to, then click the Create Forwarder button.

To delete a Forwarder delete its name or its forwarding address and click the Update button.

Updating Forwarding Address

To update a Forwarder, click the Forwarder name on the Objects list page. The Forwarder Settings page appears.

Forward to:	mark-dept@contractor.example
-------------	------------------------------

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 with a Web browser, and enter a new Forwarder name into the New Forwarder Name field. 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.

<input type="button" value="Remove Forwarder"/>	<div>New Forwarder Name: <input type="text"/></div> <input type="button" value="Rename Forwarder"/>
---	---

Removing Forwarders

If you want to remove a Forwarder, open its Settings page with a Web browser, and click the Remove Forwarder button.



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](#), [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 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 create 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 submailbox 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 submailbox. 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 CommuniGatePro 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](#), [WebUser Interface](#), [Real-Time Application](#).

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 protocol operations, and it can be set and reset explicitly with mail client applications.	\Seen	Unseen
Read	<i>same as Seen</i>		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 applications.	\$MDNSent	NoMDNSent

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. 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

Mailbox Access Control Lists

The CommuniGate Pro Server maintains an Access Control List (ACL) for every mailbox it creates. Each element of the Access Control List contains a name and a set of Mailbox access rights granted to that name.

The Access Control Lists are used to control the [Foreign Mailbox Access](#) feature that allows one Account user to access mailboxes in other Accounts.

An ACL element name can be:

`anyone`

This ACL element specifies the access rights granted to everybody.

`anyone@`

This ACL element specifies the access rights granted to everybody in the same CommuniGate Pro Domain.

`anyone@domainName`

This ACL element specifies the access rights granted to everybody in the CommuniGate Pro *domainName* Domain.

`accountName`

This ACL element specifies the access rights granted to the *accountName* Account user.

`accountName@domainName`

This ACL element specifies the access rights granted to a user in a different CommuniGate Pro Domain.

#groupName

This ACL element specifies the access rights granted to all members of the *groupName* [Group](#) (in the same Domain).

An ACL element name can have a + or a - prefix.

Account owners always have all access rights to all mailboxes 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.

A Server Administrator with the All Accounts and Domains [access right](#) has all access rights for all Server or Cluster mailboxes.

Domain Administrators with the CanViewMailboxes access right have all access rights for all mailboxes in their Domains.

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 submailboxes "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 submailbox is created, it inherits the ACL of the "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.

The Access Control Lists can be set and modified using either the WebUser Interface or using a decent [IMAP](#) client.

In order to be able 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.

In order to be able 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).

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`.

Samples:

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:

<code>anyone@</code>	Lookup, Select, Seen
<code>-john</code>	Select, Seen
<code>+susan</code>	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.

<code>anyone@company2.com</code>	Lookup, Select, Seen
<code>-john@company2.com</code>	Lookup, Select, Seen
<code>susan@company3.com</code>	Lookup, Select, Delete

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 [foreign mailbox](#) 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 TextMailbox (.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 <return-path>(flags-UID) 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 and the unique mailbox message ID).

If a mailbox file has been copied from an old system, or when it is used as an [External 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 TextMailbox mailboxes become less effective as their size grows. When a TextMailbox 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 TextMailbox 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 TextMailbox opening speed, the CommuniGate Pro can maintain a mailbox index (.idx) file alongside the TextMailbox 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 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.

You can modify the TextMailbox Manager settings by opening the Obscure page in the Settings realm of the WebAdmin Interface:

TextMailbox Manager	
Concurrently used large buffers:	<input type="text" value="7"/>
Index Mailboxes larger than:	<input type="text" value="300 Kbytes"/>

`Concurrently used large buffers`

Use this setting to specify how many concurrent operations (parsing, deletion) the TextMailbox Manager can perform on large mailboxes.

`Index Mailboxes larger than`

Use this setting to specify the minimal size for mailboxes that need indexing.

The MailDirMailbox (**.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:

iiii-flags-timestamp

where *iiii* is the message unique ID, *flags* are the message status flags, and the *timestamp* is the message *internal time stamp* - the time (GMT) when the message was added to the mailbox, in the `yyyymmdd-hhmmss` format.

Note: On the Unix platforms, the `.mdir` 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.

Note: most of 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 very 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 at the same time, and all synchronization is implemented in 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 .mbox format is more efficient than .mdir in most cases, this is why this format is used as the default one. The .mdir format is recommended only for those mailboxes that contain many (20 or more) large (100K or more) messages. If a user has a `Proposals` mailbox where she stores all messages with attached documents, each 50-70K in size, then this mailbox may work faster if it is created in the .mdir format.

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](#).

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 [foreign mailbox](#) 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) 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 IMAP clients access foreign mailboxes.

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 IMAP mailer, 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 ("`/`") sign.

Mailbox aliases can contain just the name of the foreign account (`~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 `~chief`.

When the user `assistant` connects to her account using any IMAP 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 `chief` 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.



File Storage

CommuniGate Pro [Accounts](#) contain a *File Storage* - a set of HTML, JPEG, and other files. This file set is also known as *WebFiles*.

The CommuniGate Pro [HTTP](#) module can be used to retrieve files from these Sites.

The CommuniGate Pro [FTP](#) module can be used to retrieve and update files in Personal File Sites, and the CommuniGate Pro [TFTP](#) module can be used to download files from Personal File Sites.

The [CLI API](#) can be used to manage the Account File Storage.

Only the Account owner or an Administrator can modify Account File Storage.

A File Storage can contain nested folders (file directories).

If the Account and its Domain have the [WebSite Service](#) enabled, anybody can retrieve Personal File Site files using any HTTP browser. If a file is located inside a "private" Folder, the browser user needs to supply the account password to access the file.

The total number of files and folders and the total size of all Personal File Site files is limited by the special [Account Limits](#) settings.

Personal File Sites can be used:

- to [store message attachments](#).

- to store audio prompts and other files used in [Real-Time Applications](#).

HTTP Access to a File Storage

CommuniGate Pro allows each user to be presented on the World Wide Web with a personal File 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 the Personal File Site at:

`<http://client1.com:8100/~jsmith>`

Personal File 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 File Site URLs, so Account and Domain aliases can be used in the Personal File Site URL.

Personal File Site 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 File Site of the `userA@domainB.com` Account.

File Storage must not contain any `index.wssp` file. This name is reserved for the File Site Management forms.

The home (default) page of a Personal File 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.

Private Folder

If the Account File Storage contains the folder with the name `private`, then files in that folder are available only to the Account owner and Administrators with the `CanAccessWebSites` Domain Access right.

The `private` folder can be used as a repository for any type of documents - the user can access them from anywhere, using any browser.

HTML-based Management

Users can manage their Account File Storage using any browser. There are two methods to access the Personal File Site 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 File Site:

`http://domain.dom:8100/~username/Index.wssp`

A browser will present a Login dialog box, and the user should enter her Account name and password in order to open the File Site administration page.

Server administrators with the All Domains and Accounts [Access Right](#) and Domain administrators with the `CanAccessWebSites` access right can access File Storage belonging to other users. They can use the same URL, opening the `Index.wssp` file, but they should provide their own account names and passwords.

Server and Domain administrators can access other users' File Storage 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:

Subdirectory documents			UP
Marker	File Name	Size	Modified
<input type="checkbox"/>	report.txt	488	20:52:49
<input type="checkbox"/>	myDocs	-->	
This Folder:		2	488
Totals:		5	976
Limits:		Unlimited	30720

Delete Marked

Upload File:

Rename Marked:

Create Folder:

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.

HTTP-based Management

Personal File Sites can be modified using the HTTP 1.1 PUT, DELETE, and MOVE methods. Some HTML design

tools (such as Netscape Composer) can use these methods to upload files to the server. These HTTP requests should contain the Authentication information: the Account name of the Personal File Site owner or the Account name of a Server/Domain Administrator, and the password for that Account.

FTP-based Management

Personal File Sites can be modified using the CommuniGate [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 topmost directory of the Account File Storage.

Special files

Certain Personal File Site files have special meanings.

`default.html`

This file is retrieved via HTTP when no file name is specified in the URL: `http://server:port/~username/` or `http://server:port/~username` is equivalent to `http://server:port/~username/default.html`.

The same is true for all File Storage subfolders: `http://server:port/~username/subfolder/` is the same as `http://server:port/~username/subfolder/default.html`

The name of the default file can be changed on the per-Account basis, by modifying the Account Settings.

`freebusy.vfb`

This text file contains the user Free/Busy information. When the Account Main Calendar mailbox is modified with the CommuniGate Pro [MAPI](#) module or with the [Web User Interface](#) module, the file is deleted. When the file is being accessed and it does not exist, the Server opens the Account, opens its Main Calendar mailbox, builds the Free/Busy information, and stores the information in the `free-busy.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.

Virtual Files

Virtual files are the names that can be used in the Personal File Site URLs. They do not specify actual files in the Personal File Sites, but they can be used to retrieve certain information.

`index.wssp`

This name is used for [HTML-based Personal File Site 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).



Account Data

The CommuniGate Pro server stores account information in several places. Most of the information is stored in the account service files, while some account data is grouped in the domain files.

Domain Files

For each CommuniGate Pro domain, a file directory is created in the `Domains` subdirectory inside the *Server base directory*. The directories have the same names as the domains.

For the main domain, the `Accounts` file directory is created inside the base directory.

Inside each domain file directory, a `Settings` file directory is created. This directory contains the following files:

`Access.settings`

This file has the dictionary format, and contains the names of the users that have administrative access rights to the server or to the 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 access rights. All other files can contain only the domain-level administrative access rights.

`Domain.settings`

This file contains the [domain settings](#).

RPOP.data

This file contains the information about all individual [RPOP](#) (remote POP) accounts that should be polled on behalf of the domain users.

Template.settings

This file contains the Account Template for this domain and provides the default 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.

WebUser

This directory contains HTML files that customize the [WebUser Interface](#) to the domain accounts and mailing lists.

The [Domain Administrator](#) can place HTML and other files into this directory (*publish* them) using any HTML composer application that supports the POST, DELETE, and MOVE HTTP methods.

Account Service Files

Every CommuniGate Pro account contains at least one (INBOX) mailbox file, and at least two service files. Service files have special file name extensions:

<code>.settings</code>	this dictionary file contains Account settings, including the Account Rules .
<code>.info</code>	<p>this dictionary file contains volatile Account information, such as mailbox sizes, last UIDs used in each mailbox, etc.</p> <p>Since the <code>.info</code> file is being modified rather often, the CommuniGate Pro Server is built to survive <code>.info</code> file corruptions. For example, if the mailbox last UID information is corrupted, the server rescans the mailbox and restores the correct mailbox info.</p>
<code>.dst</code>	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 .

<code>.web</code>	this optional file directory is the Account Personal File Site .
<code>.rpopids</code>	this optional file directory contains a file for each RPOP Account that has the Leave On Server option enabled. Each array file contains UIDs of the messages already retrieved from that PROP Account.
<code>.roaming</code>	this optional dictionary file contains Netscape Roaming 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.

Netscape Roaming

The [Netscape](#)® Communicator product can use any advanced HTTP server to store and restore its settings.

To use this Netscape Roaming service, the user should specify the following URL as the *Roaming Server URL*:
`http://domain[:port]/Settings/`

where *domain* is the user domain (or the main CommuniGate Pro domain), and *port* is the CommuniGate Pro [User HTTP](#) port.

The actual account name is not specified in this URL. Access to the `/Settings/` realm requires authentica-

tion, and the CommuniGate Pro Server opens the account specified in the browser username/password dialog box.

If the URL used contains the correct domain name of the target account, the account name can be specified as a simple name (i.e. without the domain part), but if the URL contains the name of some other CommuniGatePro domain (because the target domain does not have any A-record), the account name should be specified along with the domain name, i.e. instead of the `jsmith` string, the `jsmith@domain.com` string should be used in the Netscape username/password dialog box.

The Netscape Settings are stored as separate files inside the account service directory with the `.roaming` extension.



Message Transfer

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.

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`.

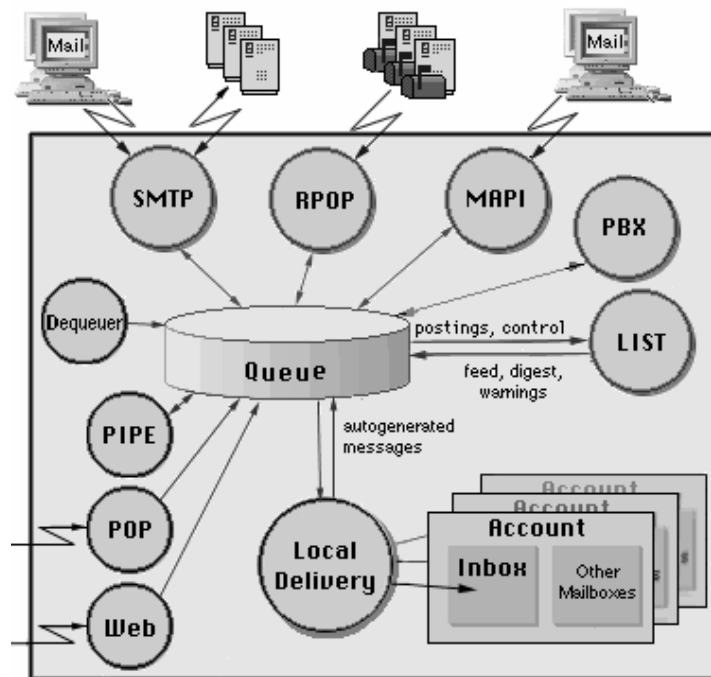
- The [PIPE](#) module submits messages received from external applications via interprocess communication channels, and the messages generated and stored in the special Submitted directory. The Submit Address tag used: PIPE.
- The [POP](#) module submits messages received from certain mailer applications 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 [Web User Interface](#) module submits messages composed within Web browsers. The Submit Address tag used: HTTP.
- The [Real-Time Applications](#) submit messages such as incoming voicemail. The Submit Address tag used: PBX.
- The [Enqueueur](#) 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 [Web User Interface](#) module generates and submits 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: LSTN.
- 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 Handler](#) 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 module 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 a systems restarts, a kernel module checks all files with the `.msg` extension stored in the `Queue` directory and resubmits them for processing.

You can use the WebAdmin Interface to configure the Temporary Files manager. Open the `Obscure` page in the `Settings` section:

Temp File Manager	
Log:	<div>Failures</div> <div>▼</div>
Recycle Temp Files:	<div>Enabled</div> <div>▼</div>

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.

To specify the Queue processing options, open the `Queue` page in the Settings section of the WebAdmin Interface:

Message Queue			
Log:	Low Level		
Queue 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 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.

You can use a Web browser to configure the Enqueuer component. Open the [Queue](#) page in the Settings section of the WebAdmin Interface.

Message Enqueuer			
Log:	<div>Low Level</div>	Processors:	<div>1</div>
Hop Counter Limit:	<div>20</div>		

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 `ENQUEUEUR` tag.

The records created when applying the Server-Wide Rules are marked with the `ENQUEUEERRULES` tag.

Processors

Use this setting to specify the number of Enqueuer processors (threads). Usually one Enqueuer thread is

enough even for a heavy-loaded server. You should increase the number of Enqueuer processors if:

- you have specified many Server-Wide [Rules](#);
- Server-Wide Rules use the `Execute` action to start external programs;
- an [External Filter](#) (anti-virus, content filtering, etc.) program is enabled.
- you have to handle heavy incoming load, with many messages being submitted every second via the SMTP or other modules.

The `numEnqueuerMessages` [SNMP](#) 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).

Dequeuing

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 kernel 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 module 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.

You can use a Web browser to configure the Dequeuer component. Open the `Queue` page in the WebAdmin Settings section.

Message Dequeuer	
Log:	Major & Failures
Processors:	1
Reporting Delay:	15 seconds
Send Return-Receipts to:	everybody
On Failure, Return:	body by default
<input type="checkbox"/> Copy Failure Reports to:	postmaster

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 `DEQUEUEUR` 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 pro-

cessing and has to generate a lot of DSN messages, should you use several Dequeueer 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 Dequeueer 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 WebAdmin, SMTP AUTH, MAPI, etc.)
nobody	positive reports are not generated.

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:

Status		Last Error			Module	Messages
Delayed till 21:03:30		some.domain.dom: no response			SMTP	3
Message	In Queue	Return Path	Addr	Size	Delayed	Last Error
26324336	2days	<>	1	1143		
26325146	29hours	<>	1	1143	24:56:12	Return-Path rejected: cannot resolve
26326340	5hours	<>	1	1143		

The table header contains the information about the entire queue ("host"):

Status

Active - this queue is being processed by the module.

Ready - this queue can be processed by the module at any time.

Delayed till time - this queue will not be processed 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.

Module

The link to the Monitor page of the Transfer module this queue belongs to.

Messages

The number of messages in this queue.

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.

Addrs

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

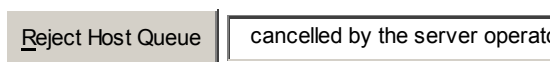
The message size.

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:



The specified text is used to generate error reports for all messages placed into this queue.

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:

Return-Path: <CGatePro-report@mail.stalker.com> Message-ID: <list-26325892@mail.stalker.com> Envelope-ID: Subject: Re: FormMail.pl configured for CGP Size: 37222 bytes (34018 in envelope) Submitted: Mon, 30 Dec 2002 09:16:58 -0800 (12hours ago)				
Recipient	Module	Host	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.

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.

Host

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.

Reject Message

cancelled by the server operator

Only the addresses that are not being processed can be rejected.

The specified text is used to generate error reports for all rejected recipient addresses.



Automated E-Mail Processing Rules

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 queue(s).

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. Open the Queue pages in the Settings section of the WebAdmin Interface 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*

This condition checks that the message **From** address is (or is not) equal to the specified *string*.

Sample:

This condition will be met for all messages coming from any account on any of stalker.com subdomains.



Sender [is | is not | in | not in] *string*

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 same as above, but the message **Sender**, **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:

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.

Rule Conditions

Each To or Cc	▼	in	▼	*@mycompany.com,*@mydept.mycc
---------------	---	----	---	-------------------------------

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	▼	*J. Smith*
-------------	---	----	---	------------

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:

Subject	▼	is	▼	*rgent*
---------	---	----	---	---------

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:

Message-ID	▼	is not	▼	*@*
------------	---	--------	---	-----

This condition will be met for all messages without the Message-ID flag and for messages that have Message-ID without the @ sign.

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:

Message Size	▼	greater than	▼	100K
--------------	---	--------------	---	------

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*
X-Mailing-List
```

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	▼	X-Mailer: CommuniGate*
--------------	---	--------	---	------------------------

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 if the message was received from a "trusted" source (via SMTP from a computer with the network address listed in the [Client IP Addresses](#) list), or from an "authenticated" source (via SMTP, WebUser, MAPI, POP XMIT, Rules - when the originator of the message has been authenticated).

Sample:

Source	▼	not in	▼	trusted,authenticated
--------	---	--------	---	-----------------------

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
	(<i>empty string</i>) in all other cases

Sample:

Security ▼	is not ▼	*encrypted*
------------	----------	-------------

Submit Address [is | is not | in | not in] *string*

This condition checks the message submit address. If the message was generated within the Server (by a Rule, etc.), its submit address is empty. Otherwise it contains the [name of the component](#) that received or generated the message and (separated with a space symbol) the network (IP) address the message came from.

Sample:

Submit Address ▼	is ▼	SMTP[10.0.0.*]
------------------	------	----------------

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 where 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

See the [Rules](#) section. 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 [flag\(s\)](#) for the message. Initially, the set of message flags is empty, or it contains the Media flag if the message contains some recorded media (voicemail, video-mail).

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 *mailbox name*

The message is copied to the specified Mailbox in your Account. The Mailbox should already exist.

Sample:

```
IF From is developer@partner.com
THEN
Store in DeveloperBox
Discard
```

If the mailbox name is specified as *~user_name/mailbox_name*, the message is stored in the *mailbox_name* Mailbox in the *user_name* 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).

`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.

`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 (,) sign.

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 and the Errors-To 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

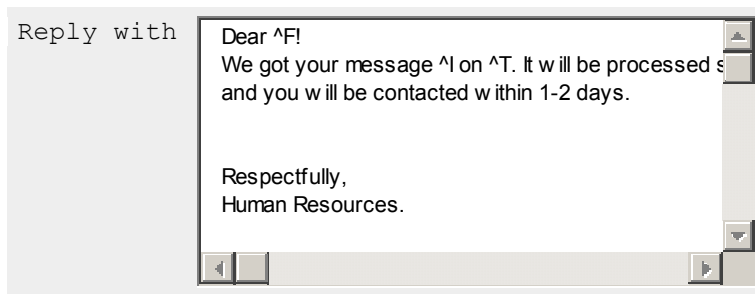
Reply-To address of the original message. If the Reply-To header is absent, the reply is sent to the From address of the original message.

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:



If the specified text starts with the '+' sign, the lines following this sign 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.

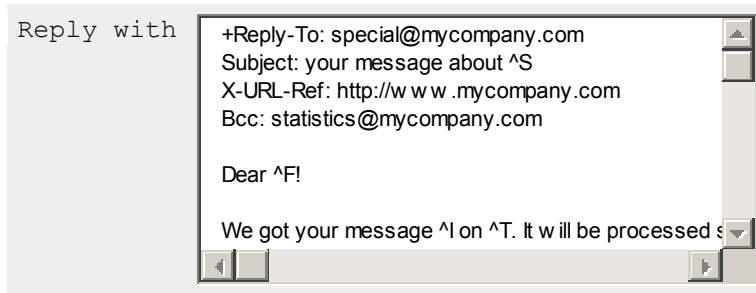
The specified header portion can contain additional To, Cc, and Bcc fields and the reply message will be sent to those addresses (the Bcc fields will be removed from the message header).

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:



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 '+' sign, the '+' sign 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*

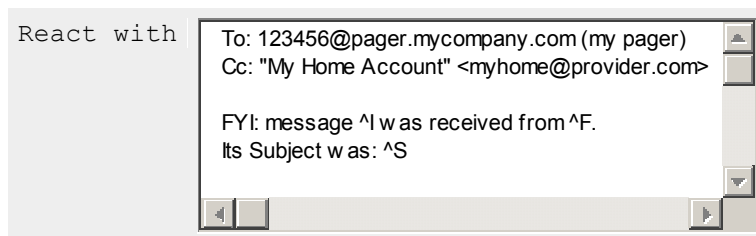
The specified message text should contain a header, an empty line, and the message body. The header should contain any number of `To`, `Cc`, and `Bcc` fields, the `Subject` field, as well as any number of additional fields.

The composed message is sent to the specified addresses.

The specified message header and the message body can contain macro symbols listed above.

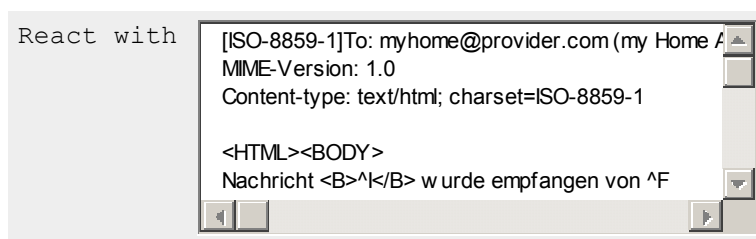
The `From`, `Sender`, `MIME-Version`, and `Content-Type` fields are composed in the same way as for the Reply With operation.

Sample:



The message text can start with the `[charsetName]` string (see above).

Sample:



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.

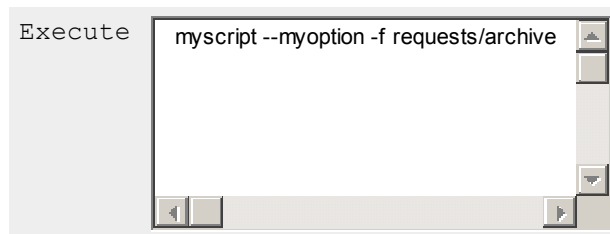
When a task is to be executed as a part of Account-Level Rule processing, the OS User Name is composed using the Account [OS User Name](#) setting, and the task is executed in that *OS User Environment*.

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:



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 parameter specifies the name of the External Filter program to use.

Sample:



Accept Request *options*

This action can be used to accept Calendaring Meeting Requests automatically. It can be used in Account-Level Rules only. See the [Calendaring](#) section for more details.

Vacation Message

Each Account has one built-in 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.

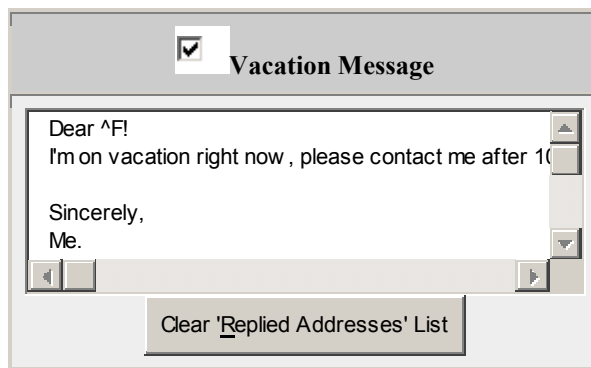
This Rule conditions are:

```
Human Generated
From           not in    #RepliedAddresses
```

The Rule actions are:

Reply with	<i>Reply Text</i>
Remember 'From' in	<i>RepliedAddresses</i>

Only the text of the Reply message can be modified:



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).

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.

Redirect All Simplified Rule

Each Account can have a simplified rule to redirect all incoming mail 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 *Redirect To* or *Mirror To*, and, optionally, *Discard*.

Only the list of redirection addresses can be modified:



☒ **Redirect All Mail to:**

myothername@other.provider.com,
thirdName@third.provider.net

☐ Keep a Copy ☒ Do not Redirect Automatic Messages ☒ Preserve To/Cc fields

Redirect All Mail to

If this option is not selected the Redirect All Rule is disabled. If this option is selected, the Redirect All Rule is enabled with the lowest priority (the rule priority is set to 1).

Keep a Copy

If this option is not selected, the action `Discard` is added to the Rule and all redirected messages are NOT stored in the account INBOX.

Do not Redirect 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.

Preserve To/Cc fields

If this option is selected, the `Mirror To` action is used for this Rule. If this option is not selected, the `Redirect To` action is used.

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.

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.



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, mes-

sages 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 [ENQUEUEUR](#) 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 Stalker Software 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 the External Filter

After you have installed an External Filter program, or built your own one, you should tell CommuniGate Pro to start that Filter Program and to establish a communication link with it. Open the General page in the Settings section of the WebAdmin Interface, and click the Helpers link. The Helpers page is displayed:

To use an External Filter program, open the General page in the Settings section of the WebAdmin Interface and click the Helpers link. The Helpers page is displayed:

Content Filtering			
<input checked="" type="checkbox"/>	Use Filter:	McAfee	
	Log:	Low Level ▼	Program Path: CGPMcAfee\CGPMcAfee.exe
	Time-out:	30 seconds ▼	Auto-Restart: 15 seconds ▼
<input checked="" type="checkbox"/>	Use Filter:	SpamCheck	
	Log:	Low Level ▼	Program Path: SpamScan\Scanner.exe -v
	Time-out:	minute ▼	Auto-Restart: minute ▼
<input type="checkbox"/>	Use Filter:		
	Log:	Problems ▼	Program Path:
	Time-out:	disabled ▼	Auto-Restart: disabled ▼

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 it into the Use Filter name field. You will use this name when you specify the External-Filter Rule conditions. 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 CommuniGatePro *base directory*, you can use the relative path (CGPMAfee\CGPMAfee.exe, for example). Otherwise, use the full path (such as D:\Programs\CGPMAfee\CGPMAfee.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.

Select the check box 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 Disabled, 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 Disabled, you need to open the Helpers WebAdmin page and click the Update button to force the Server to restart the External program.

Using the External Filter

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 checkbox next to the Filter name is selected. You may want to unselect the checkbox to let messages bypass this External Filter program. If the *filterName* checkbox is not selected, 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	2048
---	is	

Action	Parameters
ExternalFilter	McAfee

External Filters are contacted from the Server [ENQUEUEER](#) threads. Since it can take several seconds to process a large message, increase the number of ENQUEUEER processors (threads) using the Queue page in the WebAd-

min Settings section. This allows the CommuniGate Pro Server to continue message enqueueing even when a large message scan is in progress.



SMTP Module

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

The 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

To configure the SMTP module, use any Web browser to connect to the CommuniGate Pro Server, and open the SMTP page in the Settings realm. To configure the SMTP module, you should have the Can Modify Settings Server access right.

Use the Log setting to specify what kind of information the SMTP module should put in the Server Log. Usually

Log:	All Info ▼	Channels:	25 ▼
------	------------	-----------	------

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.

	Send Directly to Recipients	Default IP Address:	OS default ▼
	Forward to	mail.provider.com	
	Send AUTH:	myname	Password: *****
Channels/Host:		4 ▼	Add Channels after:
			minute ▼
Recipients/Message:		unlimited ▼	<input type="checkbox"/> Hide Received Fields
			<input type="checkbox"/> Always use 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.

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 (*) sign 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.

Send AUTH

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 Send 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`

Use this setting to specify the "reasonable wait" time period.

Additional connections to a domain mail server are opened if:

- the Server Queue for that domain contains more messages than the number of already opened channels;
- neither channel succeeds to complete a message transfer within the specified period of time or the number of messages in the domain queue is more than 50.

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 minutes
for the first:	30 minutes
then Every:	hour
Keep Trying for:	3 days
Messages with Empty Return Paths	Keep Trying for:
	2 hours
Delay Messages:	30 minutes
Delay Recipients:	hour
Send Warnings	After:
	3 hours

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.

Keep Trying

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: Keep Trying

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:

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.

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.

Log:	All Info ▼	<u>listener</u>	Channels:	25 ▼
------	------------	-----------------	-----------	------

Channels Limit

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.

Advertise:	AUTH to:	everybody ▼	8BITMIME to:	non-clients ▼
	NO-SOLICITING:	dom.spammer:ADLT,dom.listing:ADV		
Verify:	Return Paths for:	non-clients ▼	HELO for:	non-clients ▼
	Check SPF records:	Disabled ▼	Reverse Connect:	Disabled ▼

Advertise AUTH capability

If a server reports (in its initial EHLO prompt) that it supports SMTP Authentication, some mailer cli-

ents (including Netscape® Messenger 4.x) 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 never converts non-ASCII messages into the MIME form itself, 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 unneeded 8bit-to-MIME message 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 HELO and 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 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 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.

Note: the SPF 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.

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			
Size:	<input type="text" value="10 Mbytes"/>	Recipients:	<input type="text" value="25"/>
<hr/>			
Non-Client Sender:	Limited to:	<input type="text" value="10"/> recipients	per: <input type="text" value="30 seconds"/>
	Delay Prompt for:	<input type="text" value="30 seconds"/>	
	Disconnect after:	<input type="text" value="20"/> errors and	Deny Access for: <input type="text" value="15 minutes"/>
	Blocked Addresses to Remember:		<input type="text" value="300"/>

Message Size Limit

This settings tells the module to reject all incoming messages that are larger than the specified limit.

Message Recipients Limit

This settings limits the number of message recipients the module can accept. Specifying a lower value makes your server less attractive for spammers.

Non-Client Sender: Limited

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 module remembers the network addresses of disconnected senders and denies SMTP connections from those addresses during the specified period of time.

Note: the module does not remember and 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](#).

Blocked Addresses to Remember

Use this option to specify the maximum number of blocked/disconnected network addresses the Server should remember.

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			
<input type="checkbox"/>	Send Wakeups	Every: <input type="text" value="minute"/>	to: <input type="text" value="mail.provider.dom"/>
<input checked="" type="checkbox"/>	Use ATRN	login name: <input type="text" value="my.login.name"/>	password: <input type="text" value="****"/>

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 send-

ing the ATRN command.

Use `ATRN`

select this option to use the ATRN (On-demand Mail Relaying) method instead of the ETRN method.

`login 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 than the standard SMTP port 25), you should specify the port name 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 CommuniGatePro 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 `domain-Name-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.

Relaying	
Relay to any IP Address:	If Received from: <input type="text" value="clients"/> IP Address
Relay to Client IP Addresses:	If Sent to: <input type="text" value="simple"/> E-mail Address
Relay to Hosts We Backup:	<input type="text" value="Enabled"/>
Accept Wakeups from:	<input type="text" value="clients"/>
Hold Mail for Domains:	<input type="text" value="client1.com, client2.com, client3.com, client4.com, *.c"/>

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 pre-

fixes (@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 the special manner:

- 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.

For 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
S: 250 user@host sender accepted
C: RCPT TO: somebody@somehost
S: 591 Your host [10.1.1.1] is blacklisted. No mail will be accepted
C: RCPT TO: abuse@somehost
S: 591 Your host [10.1.1.1] is blacklisted. No mail will be accepted
C: RCPT TO: blacklist-admin@somehost
S: 591 Your host [10.1.1.1] is blacklisted. No mail will be accepted
....
```

**Router
Table****SMTP
protocol**

```
<abuse*@blacklisted> = postmaster

C: MAIL FROM: user@host
S: 250 user@host sender accepted
C: RCPT TO: somebody@somehost
S: 591 Your host [10.1.1.1] is blacklisted. No mail will be accepted
C: RCPT TO: abuse@somehost
S: 250 abuse@somehost@blacklisted will leave Internet
C: RCPT TO: blacklist-admin@somehost
S: 591 Your host is blacklisted. No mail will be accepted
....
```

**Router
Table**

```
<blacklist-admin*@blacklisted> = postmaster
```

**SMTP
protocol**

```

C: MAIL FROM: user@host
S: 250 user@host sender accepted
C: RCPT TO: somebody@somehost
S: 591 Your host [10.1.1.1] is blacklisted. Send your questions to black-
list-admin@mycompany.com.
C: RCPT TO: abuse@somehost
S: 591 Your host [10.1.1.1] is blacklisted. Send your questions to black-
list-admin@mycompany.com.
C: RCPT TO: blacklist-admin@somehost
S: 250 blacklist-admin@somehost@blacklisted will leave Internet
....

```

Routing

The SMTP module immediately (on the first [Router](#) call) accepts messages addresses 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` suffix is deprecated.

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. The `.smtpq` suffix is deprecated.

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.

There are two methods to include the port number into an E-mail domain:

- Use the IP-address notation: `[xx.yy.zz.tt:port]` where *xx.yy.zz.tt* is the IP address, *port* is the port number. Sample Router record:
`local.com = [192.0.0.5:26]`
- Use the `.relay` or `.via` suffix with the port number: `domain.port.via`. 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.stalker.com = mail.stalker.com.26.via`

Monitoring SMTP Activity

The Monitors realm of the [WebAdmin Interface](#) allows Server Administrators to monitor the SMTP module activity. The SMTP Monitor page contains three frames - the sending framing, the waiting frame, and the receiving frame.

The Sending frame displays the active outgoing SMTP connections:

SMTP Sending		Domains: 2					Channels:2
ID	Domain	Q	Elapsed	Status	Message	Size	Return Path
158996	domain1.dom	1	8sec	opening connection			
158997	domain2.dom	2	2sec	opening connection			



Local Delivery Module

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 a Web browser to connect to the CommuniGate Pro administrator port and open the LOCAL page in the Settings section.

Log:	Problems ▼		Processing	Processors:	10 ▼
If Mail Service is disabled: Keep for		2 days ▼	Every	hour ▼	
If the account is full: Retry for		0 seconds ▼	Every	15 minutes ▼	
Suspend:		<input checked="" type="checkbox"/> Account Queue	<input checked="" type="checkbox"/> Message		
Send Warnings after		15 minutes ▼			

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.

Suspend

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).

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.

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.

Message Flow Control

Flow Control			
Suspend Account Queue if Received	<input type="text" value="10"/>	messages within	<input type="text" value="15 seconds"/>
Suspend Account Sending if Sent	<input type="text" value="15"/>	messages within	<input type="text" value="minute"/>

Incoming 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.

Outgoing Flow Control

These settings are used to prevent system abuse by the system own users.

The Outgoing Flow control settings are applied to the messages submitted from *authenticated sources*. All messages submitted by a CommuniGate Pro Account user via SMTP using the AUTH operation, via the WebUser Interface, via the MAPI module, and via the POP module XTND XMIT command are counted. If the amount of messages submitted during the specified period of time exceeds the specified limit, the Account user's ability to submit messages is suspended.

Note: some of your system 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 system user and those messages are not counted. 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 AUTH option in the SMTP module settings.

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 sign. In this case the name of the unknown local account is used to substitute the asterisk sign.

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 to a certain local Account, if the `.local` domain suffix is used (see above).

Example:

The Router line:

`client1.com = C11.local`

All messages sent to the client1.com domain are directed to the C11 local Account.

Unified domain-wide Accounts are useful if the client systems retrieves messages from your server using the CommuniGate POP, 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 C11 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 = C11.local
```

A message is sent to:
abcdef@client1.com, xyz@client1.com

It is stored in the C11 Account, and a header field:
X-Real-To: abcdef, xyz

is added to the stored message

Note: the

```
<*@client1.com>= C11
```

Router alias record also stores all messages sent to the client1.com domain in the C11 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 that queue, opens the addressed Account, and checks if the Account has Automated Rules specified.

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:	X-Real-To
	<input type="checkbox"/> 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 sign (+) and some string. You can set this settings to:

Disabled

The Local Delivery module will not process the plus signs in Account names.

Enabled

The Local Delivery module will check for the plus sign in Account names and delete the first plus sign 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 sign in Account names and process the string after

the plus sign 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.



RPOP Module

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 Server mailboxes, the RPOP module retrieves messages from other (remote) hosts and delivers them to user mailboxes or to other destinations.

For each registered user, the RPOP module can retrieve messages 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) 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.

The RPOP module is useful even if the CommuniGate Pro Server has a full-time Internet connection. A user that has several accounts on several hosts can instruct the RPOP module to poll those accounts, so all their mail is

collected in their CommuniGate Pro account.

The RPOP module supports 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 headers. 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 a Web browser to configure the RPOP module.

Log:	<input type="text" value="Problems"/>	Polling	Channels:	<input type="text" value="10"/>
Delay Failed Hosts for:	<input type="text" value="3 minutes"/>	Default IP Address:	<input type="text" value="OS default"/>	
Delay Failed Accounts for:	<input type="text" value="30 minutes"/>	<input type="checkbox"/>	Use Domain IP Addresses	
Minimum Poll Period for Users:	<input type="text" value="10 minutes"/>	<input checked="" type="checkbox"/>	Use APOP	
Maximum Number of Accounts per User:	<input type="text" value="3"/>	<input checked="" type="checkbox"/>	Allow Self-Poll	

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.

Channels

When you specify a non-zero value for the Channels Limit 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.

Default IP Address

This option selects the default *source network address* for 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 POP3 connections made for Account-level RPOP records. If this option is selected, the RPOP module will use the first Assigned IP Address for the Domain the RPOP record belongs to.

If this option is not selected, or if the Domain does not have any Assigned IP Address, or if the RPOP record is a UDWA-record (see below), the RPOP module uses the Default IP Address as the 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 marks an account as "failed". The option specifies when the RPOP module should make the next attempt to poll the failed account.

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.

Minimum Poll Period for Users

If some users are allowed to specify their own [individual RPOP accounts](#), they may select too short Poll

Periods, generating a lot of network traffic and consuming the server resources. Use this option to set the minimum value the Server users can specify in their **Poll Every** remote account settings. This limit applies to users only. The administrator can always specify any Poll Period for the Unified Accounts and for individual RPOP accounts.

Maximum Number of Accounts per User

If some users are allowed to specify their own [individual RPOP accounts](#), they may specify too many accounts, generating a lot of network traffic and consuming the server resources. Use this option to limit the number of RPOP accounts the Server users can specify. This limit applies to users only. The administrator can always specify any number of Unified and individual RPOP accounts.

Click the Update button to modify the RPOP module settings.

Specifying Unified Domain-Wide Accounts

If a mail account on an external host collects mail directed to all users of your domain, the RPOP module can be instructed to retrieve mail from that account and distribute it to local users.

Unified Domain-wide Accounts							
Poll Every	Account	at Host	Password	Leave APOP	TLS	Special Header	
2 minutes ▼	rpoptest	node10.stalker.c	*****	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	X-Real-To
----- ▼				<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Poll Every

This option specifies how often the RPOP module should poll the remote account. Set this option to -- -- to remove the remote account record. If you set this option to **disabled**, the account record is not removed, but the remote account is not polled.

Account

This option specifies the name of the mail account on the remote host. For Unified Domain-Wide Accounts, this name is usually your domain name or part of your domain name.

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.

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 account 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.

APOP

If this option is selected AND the UseAPOP 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.

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
```

Special Header

The name of the messages header (RFC822) field that the provider host inserts into the messages stored in the Unified Domain-Wide Account (see below).

There is always an empty row in the Unified Accounts table. Use it to specify a new Unified Account. To remove an account, set the Poll Every option to -----.

Click the Update button to modify the RPOP module list of the Unified Domain-Wide Accounts.

Special Headers and Mail Distribution

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 the protocol commands, if mail is sent via UUCP, the envelope is sent using additional files. The information in the envelope is usually the same as the information in the message headers, but it is not always true. The most important exceptions are:

- the message headers do not contain the addresses of the Bcc recipients
- the headers of a mailing-list message do not contain the addresses of the mailing list subscribers.

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 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.

All Stalker mail servers can be used to provide Unified Domain-Wide Accounts. For 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.

Mail Distribution without Special Headers

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, leave the Special Header field empty.

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 envelope information to the messages stored in your Unified Domain-Wide Account, so you can use the Special Header feature.

Specifying Remote Accounts for Individual Users

The CommuniGate Pro RPOP module can poll POP accounts on remote hosts on behalf of the CommuniGate Pro users (Accounts). For each CommuniGate Pro user several external POP accounts can be specified. External accounts can be specified by the Server administrator, via a link on the [Account settings](#) page, or by the users themselves, via the WebUser Interface, if the right to specify remote POP accounts has been granted to the user.

Poll Every	Account	at Host	Password	Leave	APOP	TLS	Last
30 minutes	mydomain	provider.com	****	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	12:34:56
----				<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

The settings are the same as for the [Unified Accounts](#), but the Special Header field is not presented. All messages retrieved on the user behalf are directed to that user, regardless of the message header contents.

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.

All messages retrieved for individual CommuniGate Pro Accounts are sent to those Accounts via the CommuniGate Pro [Queue](#), so all Server-Wide and Account-Level [Rules](#) are applied to those messages.

All messages retrieved for individual CommuniGate Pro accounts get the 'do not report failure' flags, so if delivery was unsuccessful, no error report is sent to the original message sender.

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](#).

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

To configure the LIST module, use any Web browser to connect to the CommuniGate Pro Server, and open the LIST page in the Setting section. To open pages in the Settings section, you should have the Can Modify Settings access right.



The screenshot shows a configuration bar with a light gray background. On the left, the label "Log:" is followed by a dropdown menu currently displaying "Problems". In the center, the word "Processing" is displayed in a bold, black font. On the right, the label "Processors:" is followed by a dropdown menu currently displaying the number "3".

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.stalker.com	ali
CGatePro	test.stalker.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. Open the [Account Settings](#) page, enter the name of the Mailing List to create and click the New List button.

The Server checks that there is no Account, Group, Forwarder or Mailing List with the same name in the same Domain, and creates a new Mailing List.

Several mailboxes are created in the list owner Account:

<i>listname</i>	this mailbox is the mailing list archive: it contains the messages posted on the mailing list
<i>listname/requests</i>	this mailbox contains the messages with subscription requests
<i>listname/reports</i>	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 list owner should redirect them back to the mailing list using a secure submit method: the CommuniGate Pro [Web User](#) 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 Accounts and Domains](#) 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/folders also provides links to the Account Mailing Lists pages

Log:	<input type="text" value="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:	<input type="text" value="R&D Discussion"/>
Preferred Character Set:	<input type="text" value="ISO-8859-1"/> ▼
Digesting and Archiving:	<input type="text" value="Enabled"/> ▼ Settings
Verify Owner Using:	<input type="text" value="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 authentication;
- 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 Accounts and Domains](#) 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
All Texts	^N	the <i>listname</i> string
	^D	the <i>domain</i> string
	^E	the Description setting
Digest Subject, Header, Trailer	^X	the sequence number of the current digest
Feed Subject Prefix, Header, Trailer	^B	the sequence number of the current mes- sage

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.

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
<i>listname-on@domain</i> or <i>listname-subscribe@domain</i>	to subscribe to the list in the default mode	to confirm the current subscription mode
<i>listname-feed@domain</i>	to subscribe to the list in the FEED mode	to change the subscription mode to FEED
<i>listname-digest@domain</i>	to subscribe to the list in the DIGEST mode	to change the subscription mode to DIGEST
<i>listname-index@domain</i>	to subscribe to the list in the INDEX mode	to change the subscription mode to INDEX
<i>listname-null@domain</i>	to subscribe to the list in the NULL mode	to change the subscription mode to NULL
<i>listname-off@domain</i> or <i>listname-unsubscribe@domain</i>		to unsubscribe from the list
<i>listname-confirm@domain</i>		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:	<input type="text" value="anybody"/> Save <input type="text" value="rejected"/> Requests
Default Mode:	<input type="text" value="feed"/> <input type="checkbox"/> Request Confirmations
Confirmation Request Message	
Subject:	<input type="text" value="Confirmation Request (^)"/>
Text:	<div>This is an automated message from the <^N@^D> mailing list manager.</div>

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 <i>listname/requests</i> mailbox (see below) and should be approved by the list owner. The List owner should redirect these requests to the <i>listname-on</i> , <i>listname-feed</i> 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 <i>listname/requests</i> 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 and subscribe operations and subscribe(<i>operation</i>) for other operations
^A	the subscriber address
^I	the confirmation identifier

Welcome and Good Bye Messages

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

Welcome/Policy Message	
Subject:	Welcome!
Text:	This is an automated message from the <^N@^D> m You have joined the ^E as <^A>
Good Bye Message	
Subject:	Good Bye!
Text:	Your address <^A> has been removed from the <^N Thank you for being w ith us, and please come back

Posting Messages

To post a message on the mailing list, the author should send it to the *listname@domain* address.

Posting Policy			
Accept Postings:	<div>from subscribers</div>	New Subscribers:	<div>Moderate first 2</div>
Allowed Format:	<div>text only</div>	Maximum Size:	<div>30 Kbytes</div>
Prohibit:	<div><input type="checkbox"/> Non-matching Character Sets</div>		
	<div><input checked="" type="checkbox"/> Unmodified Digest Subjects</div>		
Service Fields:	<div><div>*auto*</div></div>		

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 secure method) will be posted
moderated	<p>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.</p> <p>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 <code>from subscribers</code> mode, the New Subscribers setting (see below) provides more advanced moderation options.</p>

<code>from subscribers</code>	Messages from the list subscribers are accepted for posting; some messages can be moderated (see below).
<code>moderate guests</code>	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.
<code>from anybody</code>	Messages from any address are accepted for posting.

New Subscribers

This settings 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 settings specifies the allowed MIME format for postings.

<code>plain text only</code>	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 part and the same text as a image/gif part/variant)
anything	messages in any MIME format can be posted

The list owner can always post messages in any format.

Maximum Size

This settings 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 Subjects

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 (*) sign, 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 <i>listname</i> string
^D	the <i>domain</i> 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

Hide 'From' Addresses

When this option is selected, the 'From' address of the posted messages is modified, so it contains the list address instead of the message author address. If the original 'From' address contained a comment (a real name of the message author), it is preserved.

The 'From' addresses in the message archives, digests, and indices are converted, too.

You can use this option to 'hide' the real E-mail addresses of those people who post messages on this mailing list.

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 and Cc:	remove	X-Mailer	

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 (*) sign into the first field.

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:	<input checked="" type="checkbox"/> suspend subscription for: 2 days
	unsubscribe after: 10 bounces
	<input checked="" type="checkbox"/> suspend till confirmation
	unsubscribe after: week
Process a Fatal Bounce as:	10 Non-Fatal
When Unsubscribing:	<input checked="" type="checkbox"/> Notify Owner
Cleanup List every:	day
Save	unprocessed Bounce Reports
Warning Message	
Subject:	WARNING!
Text:	This is an automated message from the <N@^D> mailing list manager

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.

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:

☒ insert after Reply Prefix

Header

Trailer

This message is sent to you because you are subscribed to the mailing list <^N@^D>.

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*).

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 (insert after Reply Prefix is not selected)	Subject: [R&D] Re: test
the distributed reply (insert after Reply Prefix is selected)	Subject: Re: [R&D] test
the composed digest	1) test 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 not inserted, and user replies are directed to the From address of a distributed messages, i.e. to 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 indices.

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:	day		
or if Larger than:	100 Kbytes	or if has:	100 messages
First Digest at:	05:30		

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:	<input type="text" value="^N Digest #^X"/>
Body Format:	<input type="text" value="plain text"/>
Header <input type="text" value="^E Digest #^X"/>	
Index Line <input type="text" value="^X02) ^S80
by ^F"/>	
Index Trailer <input type="text" value="This digest is sent to you because you are subscrib
the mailing list <^N@^D>.
To unsubscribe, send any message to: <^N-off@^D"/>	
Trailer <input type="text"/>	

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 signs.

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			
Maximum Archive Size:	300 Kbytes	Messages to Keep:	1000 messages
Start New Archive every:	month	Who can Browse:	subscribers

Maximum Archive Size

This settings 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 Server Web 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. All users trying to view the list from outside the specified addresses/networks this mode works as the subscribers mode and they have to enter the name/password pair (E-mail and Confirmation 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.

Display	first	1000	▼	Filter:	<input type="text" value=".stalker."/>													
2 of 289 selected																		
	E-mail Address	Mode	Subscr	Posts	Bounces	Real Name												
<input type="checkbox"/>	andy@vax.stalker.com	null	15:54:51	3		Andy												
<input type="checkbox"/>	test@mail.stalker.org	feed	18:18:11	2 mod		Test Account												
<table><tr><td>Mark All</td><td>Unsubscribe</td><td>Set</td><td>Unmoderated</td><td>▼</td><td>postings</td></tr><tr><td></td><td>Mark Failed</td><td>Set</td><td>feed</td><td>▼</td><td>mode</td></tr></table>							Mark All	Unsubscribe	Set	Unmoderated	▼	postings		Mark Failed	Set	feed	▼	mode
Mark All	Unsubscribe	Set	Unmoderated	▼	postings													
	Mark Failed	Set	feed	▼	mode													

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

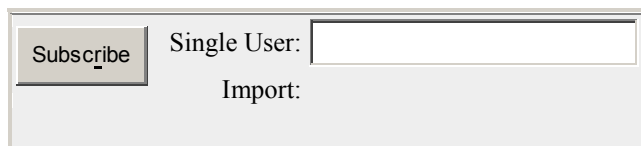
These controls allow you to change the moderation mode for the selected users. You can select some subscribers and set their posting mode to moderated, prohibited, unmoderated, or special. See the [Posting Messages](#) section for more details.

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.




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 subscribers 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

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

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 multipart/alternative format containing a part in the plain text format.

The LIST module routes to itself all addresses in the form

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

on, off, subscribe, unsubscribe,
feed, digest, index

report

anything else

Action

messages sent to these addresses are processed as [subscription requests](#).

messages sent to these addresses are processed as delivery reports about the distributed messages

messages are rejected



PIPE Module

The LIST module processes messages sent to `listserver@localdomainname`. The module takes the List Server commands. 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 `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 signs, 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 a Web browser to open the PIPE page in the Settings realm of the WebAdmin Interface.

Log:	<input type="text" value="Problems"/>	Delivering	Processors:	<input type="text" value="3"/>
Application Directory:		<input type="text"/>		
Processing Time Limit:		<input type="text" value="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 Submitted Directory Every:	10 seconds ▼

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*.



Real Time Signals

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 Module, internal sessions, and "call legs", internal kernel components, etc.). The Server either processes (terminates) these Requests itself, or it delivers them to remote entities (via the SIP 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 SIP Server when they go on-line. The Server registers the users by remembering their "SIP identifier" and the network (IP) addresses they use.

Each user has a unique "SIP identifier", also called AOR (Address of Record).

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, [Forwards](#), and other Routing methods work for Real-Time Communications, too.

Signals

A Signal is a basic Real-time Task. One Real-time entity sends 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.
- Internal [Real-time Node](#), such as [Real-Time Application](#) nodes (such as PBX application) can send call various Signal requests.
- [Automated Processing Rules](#) can use Signals (to send Instant Messages as notifications).

Processing Requests

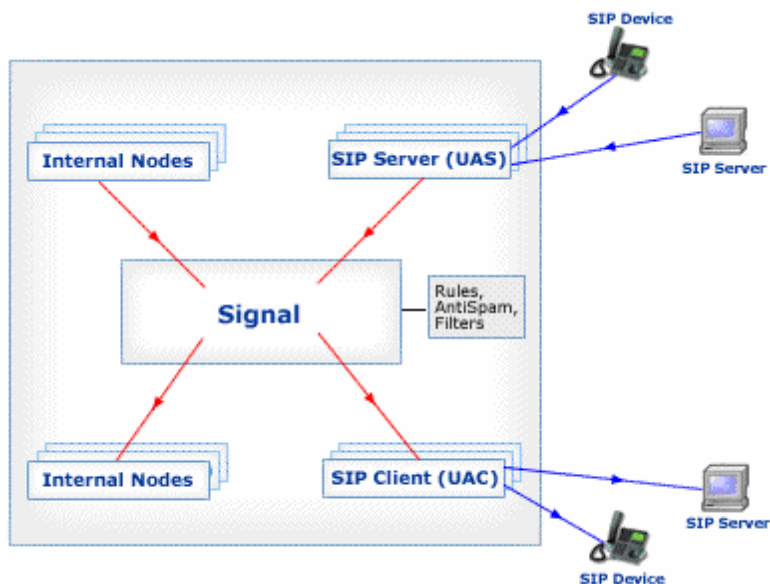
When the Signal Module receives a Request, it calculates the target URI for it. It takes the Request URI (or the first Route URI in the Request Route set), and uses the [Router](#) component to detect the Request destination.

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 the [Forking process](#) starts with the Request URI as the initial AOR set.

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 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 cancelled.

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 `RealTime` page in the `Settings` section:

Log:	Failures ▼	Processing	Processors:	3 ▼
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.

Limit

This setting specifies how many Signal "tasks" the component can handle at the same time. If this limit is exceeded, new Signals are rejected, and an error Response is sent to the Signal sender.

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.

Sending to an Account

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 Real-Time settings are retrieved along with the Account registration. These Real-Time settings modify the generic Forking algorithm.

To specify these settings, open the Account settings page using the WebAdmin Interface, and follow the Real-Time link.

An Account user can also modify these settings using the [WebUser Interface](#). See the [PBX](#) section for more details.

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.

Authentication

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 [SIP Module](#) implements the BASIC, DIGEST, and NTLM authentication methods.

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 SIP page in the WebAdmin Settings realm.

Services		
Registration: Minimal:	<input type="text" value="30 seconds"/>	Default: <input type="text" value="30 minutes"/> Maximal: <input type="text" value="week"/>

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.

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.

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.

Presence

The Signal Module implements a Presence Server. The Module supports the `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.

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 unread 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.

Subscription to the Registration package is available to the Account owner, and to other Accounts with the `Can-Impersonate` [Access Right](#).

Local Nodes

The CommuniGate Pro Server dynamically creates various 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 Signalling functions:

- The [Real-Time Application](#) component creates a Node for each "call leg".
- The [MAPI](#) and [WebUser](#) modules create Real-Time Nodes to implement the "Make Call" function. The Node these modules create sends Requests to establish a call between the session user and the specified address.
- The SIP Module creates a Node and uses it to send Register Requests to External Gateways to maintain the Server Registration on those Gateways.
- 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 *RealTime* page in the *Settings* section, and follow the Nodes link:

Log:	Failures ▼	Processing	Processors:	3 ▼
Nodes 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 *SIGNODE* tag.

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.

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.

Call Detail Records (CDRs)

The Signal module can generate Call Detail Records for INVITE and BYE transactions it processes.

CDRs can be generated and stored in CDR Log files. CDR Log files are text files, with each record stored as a separate line. Each line has the following format:

```
hh:mm:ss.ddd cdr_data
```

where *hh* is the hour, *mm* is the minute, *ss* is the second, and *ddd* is the millisecond of the moment when the CDR record was generated, and the *cdr_data* is the CDR data.

Monitors	
<input checked="" type="checkbox"/>	Record CDRs

Record CDRs

Select this option to generate CDR Log files.

When the [External CDR Processor](#) Helper application is enabled, the Signal Module generates CDRs and sends them to that application.

CDR data text has the following format:

```
01 NNN-method <callID><fromTag><toTag> <from><to> [reqIP] [respIP] flags
```

where:

- 01
the CDR format version number
- NNN
the transaction result code (numeric)
- Method
the transaction operation/method (INVITE, BYE).
- callID, fromTag, toTag,
the dialog Identifier (the Call-ID field, the From field tag parameter, the To field tag parameter).
- 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.

from, to

the transaction `From` and `To` field URIs.

reqIP

the network (IP) address the transaction request was received from.

respIP

the network (IP) address the transaction response was received from.

flags

optional elements, separated with space symbols:

[`auth:accountName`]

this element is added if the transaction request is authenticated. The *accountName* is the name of the authenticated CommuniGate Pro Account.



Automated Signal Processing Rules

The CommuniGate Pro Server can automatically process Signals using sets of [Automated Rules](#). Only the Signal Request sent outside already established Dialogs are subject to Rules.

The Server-Wide and Cluster-wide Rules are applied to all Signals sent to the Server and to the Cluster.

When a Signal 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.

Specifying Signal Rules

System administrators can specify Server-Wide and Cluster-wide Signal Rules. Open the Signals pages in the Settings section of the WebAdmin Interface 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 Set-](#)

tings pages.

See the [Automated Rules](#) section to learn how to set the 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:

Method [is | is not | in | not in] *string*

This condition checks if the Request Method address is (or is not) equal to the specified *string*.

Sample:

Method ▼	is ▼	INVITE
----------	------	--------

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 address 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 ▼	AV
------------	------	----

This condition will be met for all audio/video INVITE Requests.

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	▼	*@*stalker.com
------	---	----	---	----------------

This condition will be met for all signals coming from any account on any of stalker.com subdomains.

To [is | is not | in | not in] *string*

The same as above, but the Request To 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	▼	*J. Smith*
-------------	---	----	---	------------

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	▼	*@*stalker.com
---------------	---	----	---	----------------

This condition will be met for all signals coming from any account on any of stalker.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:

Presence	▼	in	▼	busy,
----------	---	----	---	-------

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

See the [Rules](#) section. The Signal is sent to its current target set.

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:

Method	▼	is	▼	INVITE
Authenticated	▼	is not	▼	adManager@domain.com

Redirect to	▼	sip:adManager@domain.com
-------------	---	--------------------------

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 advertizing 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.

CPL Scripts

You can create and use CPL scripts to control Signal Processing.

CG/PL Programs

You can create and use [CG/PL](#) programs to control Signal Processing.

Logging Rules Activity

The [Signal](#) component records Server-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.



SIP Module

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 actual media data, too.

SIP Server Settings

To configure the SIP module, use a Web browser to connect to the CommuniGate Pro Server WebAdmin Interface, and open the SIP page in the Settings realm.

To configure the SIP module, you should have the Can Modify Settings [access right](#).

Click the Receiving link to open the SIP Server (UAS) Settings.

Transport

The Transport panel allows you to configure the network-level options for SIP packet receiving:

Transport		Log: Major & Failures
UDP	listener	
TCP	listener	Input Channels: 5
		Idle Timeout: 10 minutes

Log

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.

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.

Transactions

The Transactions panel allows you to specify how the SIP Module handles SIP server (UAS) transactions.

Server Transactions		Log: Problems
Limit: 300	Processors: 5	

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.

Limit

Use this setting to specify the maximum number of concurrent server transactions the SIP Module is allowed to handle.

Processors

Use this setting to specify the number of threads used to process SIP Server transactions.

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	
<input checked="" type="checkbox"/> Advertise Digest AUTH	<input checked="" type="checkbox"/> Advertise NTLM AUTH

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.

SIP Client Settings

To configure the SIP Client (UAC) settings, use the WebAdmin Interface to open the SIP Module Settings pages and click the Sending link.

Transport

The Transport panel allows you to configure the network-level options for SIP packet sending:

Transport		Log: <div>Problems</div>
UDP Request Size Limit	LAN:	<div>20K</div>
	WAN:	<div>1300</div>

Log

Use this setting to specify the type of information about SIP packets and SIP transport the module should put in the Server Log.

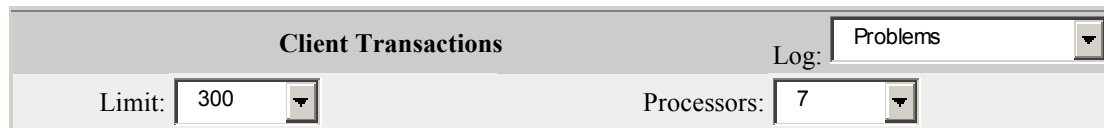
This is the same settings as the Transport Log Level setting displayed on the SIP Server settings page.

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.

Transactions

The Transactions panel allows you to specify how the SIP Module handles SIP client (UAS) transactions.



The screenshot shows a configuration panel titled "Client Transactions". It contains three settings: "Log:" with a dropdown menu set to "Problems", "Limit:" with a text box containing "300" and a dropdown arrow, and "Processors:" with a text box containing "7" and a dropdown arrow.

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.

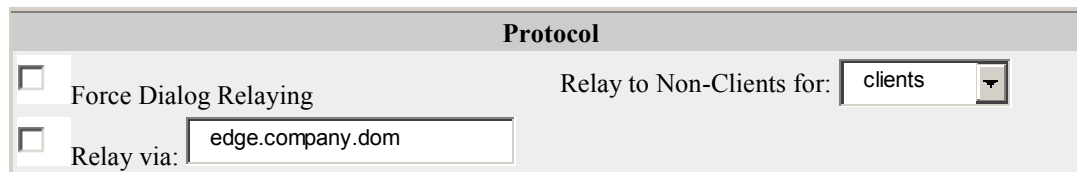
Limit

Use this setting to specify the maximum number of concurrent client transactions the SIP Module is allowed to handle.

Processors

Use this setting to specify the number of threads used to process SIP Client transactions.

Protocol



The screenshot shows a configuration panel titled "Protocol". It contains three settings: "Force Dialog Relaying" with an unchecked checkbox, "Relay to Non-Clients for:" with a dropdown menu set to "clients", and "Relay via:" with a text box containing "edge.company.dom".

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.

External Gateways

You may want to use your CommuniGate Pro Server with external SIP Gateways. External Gateways can be used to relay calls to the public telephony network (PSTN) and to relay PSTN calls to your Server. These services are usually fee-based and communications with External Gateways may require authentication.

You can also use the External Gateways features if you need to communicate with remote SIP networks using authentication.

Note: if you use an in-house PSTN gateway, it is usually configured to accept all calls from within your LAN (so your Server does not need to use Authentication to relay call Signals to that gateway), and that gateway can be configured to direct incoming connections to a certain CommuniGate Pro Server Account (so there is no need to

register your Server with that gateway). In this case you can employ this PSTN gateway without using the External Gateways features.

Use the External Gateway panel to specify the Gateway settings:

Gateway Name:	<input type="text" value="Provider1"/>	Calls:	Authenticate:	<input type="text"/>	<input type="button" value="v"/>
			Substitute:	<input checked="" type="checkbox"/>	From <input type="checkbox"/> To
Username:	<input type="text" value="4153837164"/>	Domain:	<input type="text" value="sip.provider1"/>	Via:	<input type="text" value="sip:10.12.34."/>
Auth Name:	<input type="text"/>	Password:	<input type="text" value="*****"/>		
Every:	<input type="text" value="5 minutes"/>	Contact:	<input type="text" value="incoming@thisServer.dom"/>		

Gateway Name

Use this setting to assign an internal name to this External Gateway. You will use that name in the [Router](#) records to address certain calls to this Gateway. Assign a unique alphanumeric name to each External Gateway you will use.

Domain

Use this setting to specify the External Gateway domain name.

Via

Use this setting if the requests for this External Gateway should not be sent to the network addresses associated with the Gateway Domain name (for example, requests sent to this Gateway should be sent via some proxy).

This setting allows you to specify an alternative domain (DNS) name, an IP address, or a complete SIP URI.

Username

Use this setting to specify a name to be used to register on and (optionally) to call via this External Gateway.

If the value does not contain the @ symbol, the Domain setting is added to form a qualified username.

Password

Use these settings to specify the password to use with the External Gateway.

AuthName

Specify this setting only when the Authentication name to be used with the External Gateway is not the same as the UserName.

Register every

Use this setting to specify how often the SIP Module should send REGISTER Requests to this External Gateway.

Contact

Use this setting to specify the address to be registered with the External Gateway. When a gateway receives an incoming call, it sends the call Signal request to the specified address.

This Contact address usually is an Account on your Server that employs an "auto-attendant" [Real-Time Application](#) to answer these calls and to direct them to other Accounts on your Server.

Calls: Authenticate

When this option is not disabled, an authentication header field (`Authorization` or `Proxy-Authorization`) is added to all call (`INVITE`) requests sent to this External Gateway. The Module employs the authentication "nonce" used for the last REGISTER request sent to the Gateway, so make sure your Register Every time period is shorter than the nonce "Time To Live" period of this External Gateway.

Calls: Substitute From

When this option is selected, the From addresses (the caller name and address) of call Signal (`INVITE`) requests are substituted with the Username (*caller masquerading*).

Calls: Substitute To

When this option is selected, the To addresses (the caller name and address) of call (`INVITE`) requests are substituted with the Signal request URI (after removing URI port number and URI parameters).

To create a new External Gateway entry, enter the Gateway Name in the last, empty table element, and click the Update button.

To remove an External Gateway entry, enter an empty string into its Gateway Name field, and click the Update button.

In order to send Signal requests to the External Gateway, you need to specify Router records.

Sample 1.

All calls to `1number@main_domain` should be directed to the `Provider1` Gateway as requests to call *number*

All calls to `+1number@main_domain` should be directed to the `Provider1` Gateway as requests to call *number*

all calls to `+number@main_domain` (where *number* does not start with 1) should be directed to the

IntlProvider Gateway as requests to call *011number*

all calls to *011number@main_domain* should be directed to the IntlProvider Gateway as requests to call *011number*

all calls to *9number@main_domain* should be directed to the LocProvider Gateway as requests to call *415number*

Use the following Router records:

```
NoRelay:Signal:<1*>    = *@Provider1.sipgw
```

```
NoRelay:Signal:<+1*>   = *@Provider1.sipgw
```

```
NoRelay:Signal:<+*>    = 011*@IntlProvider.sipgw
```

```
NoRelay:Signal:<011*>  = 011*@IntlProvider.sipgw
```

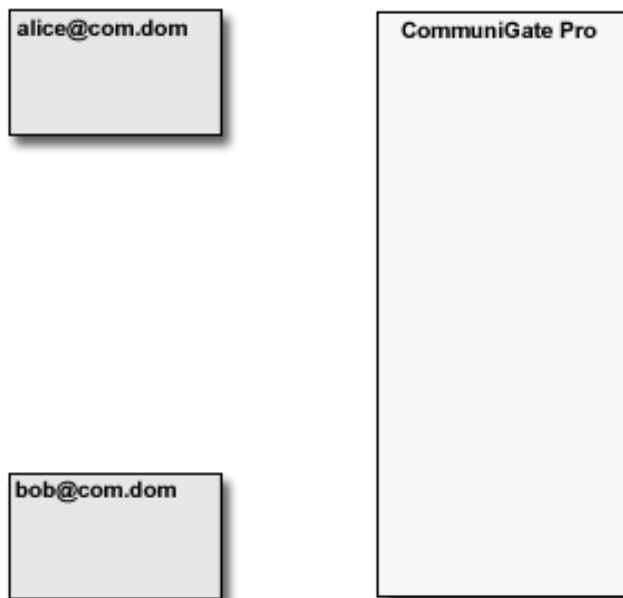
```
NoRelay:Signal:<9*>    = 415*@LocProvider.sipgw
```

Note: you should use the NoRelay: prefix to avoid turning your Server into an open relay that would allow any external user to relay calls to the External Gateway using your Server credentials.

NAT Traversal and Media Stream Proxy

The "original, "basic" SIP communication model assumes that endpoints can communicate directly, i.e. that all "elements", including SIP clients - phones, softphones, PBX applications), and SIP servers have "real" Internet IP Addresses. In this situation the Server (SIP Proxy) is needed only to establish a call. Media data and in-call signalling requests are sent directly between the endpoints:

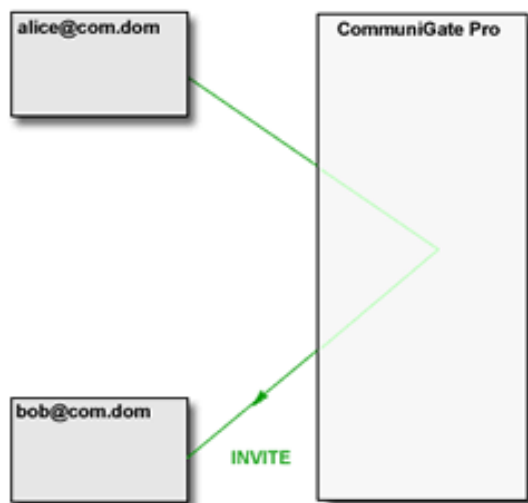
Step 1.



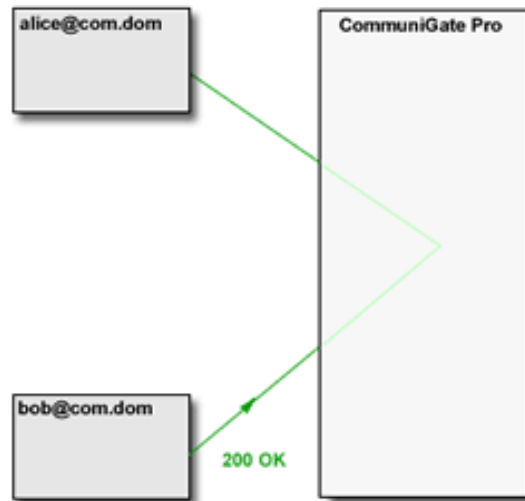
Step 2.



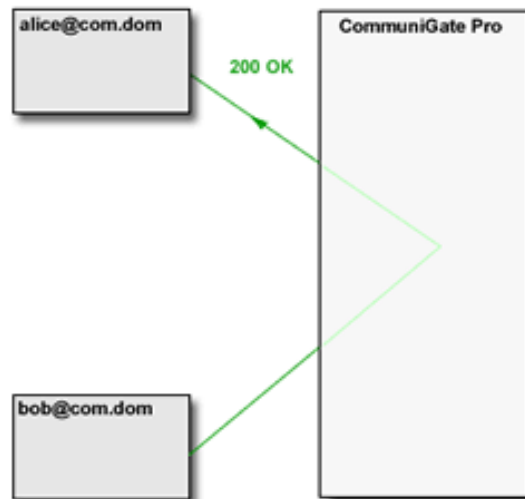
Step 3.



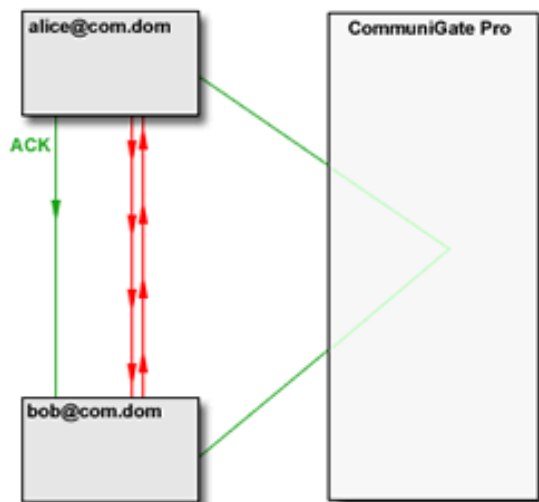
Step 4.



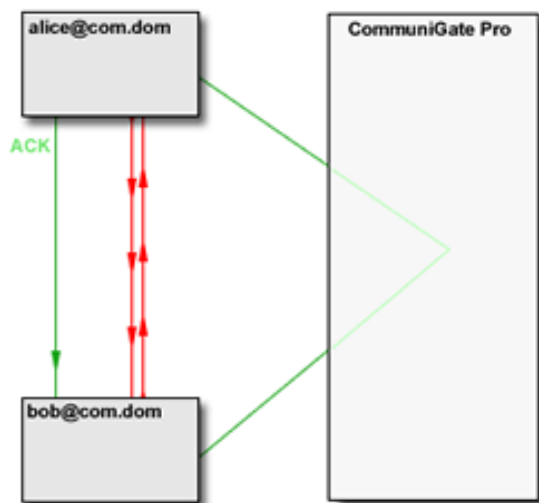
Step 5.



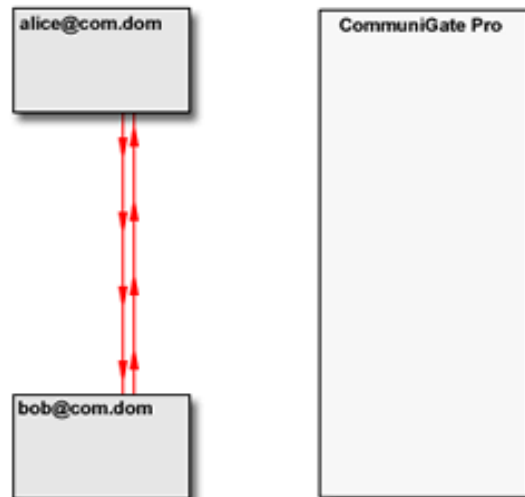
Step 6.



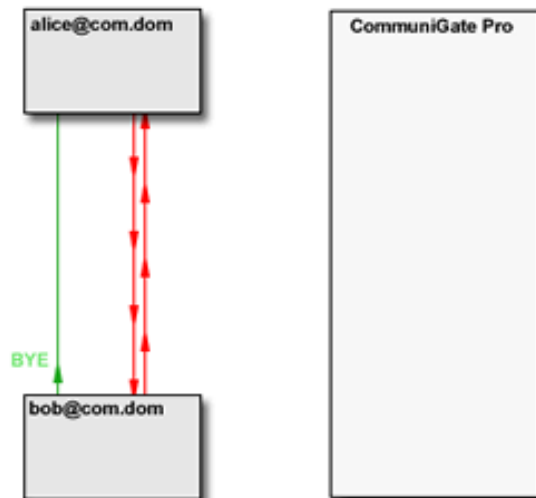
Step 7.



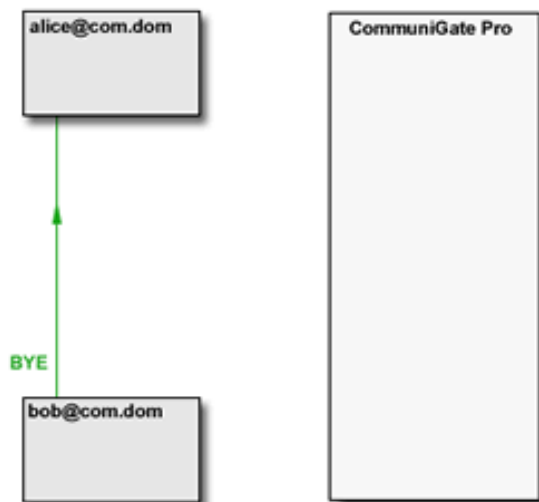
Step 8.



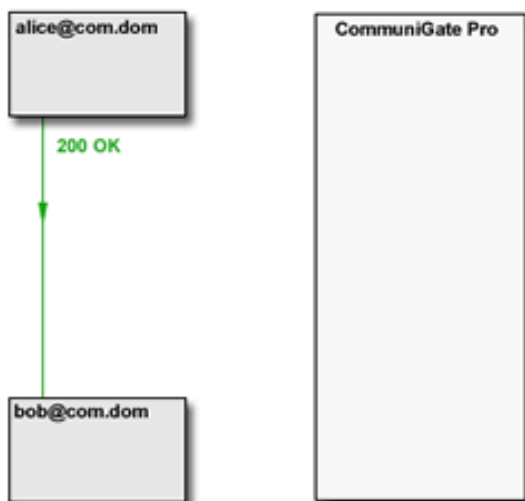
Step 9.



Step 10.



Step 11.

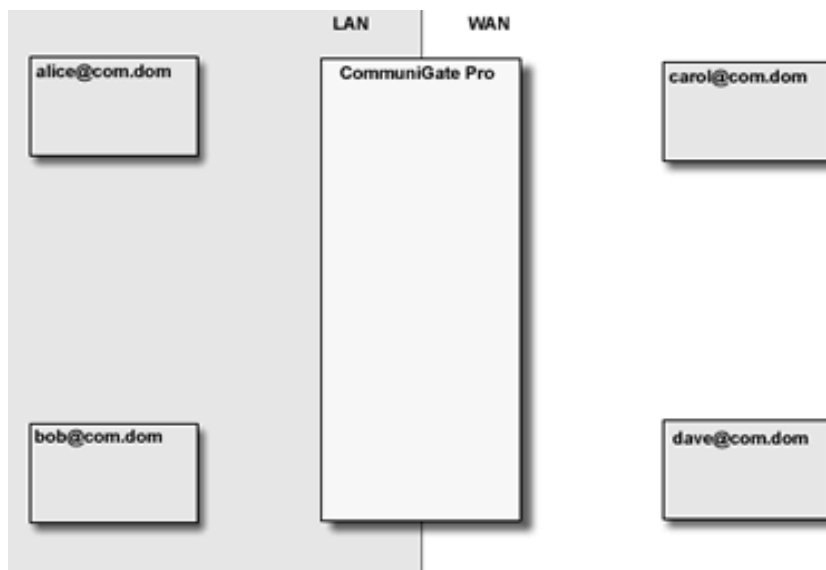


CommuniGate Pro supports automatic "NAT traversal" for the standard-based real-time communications.

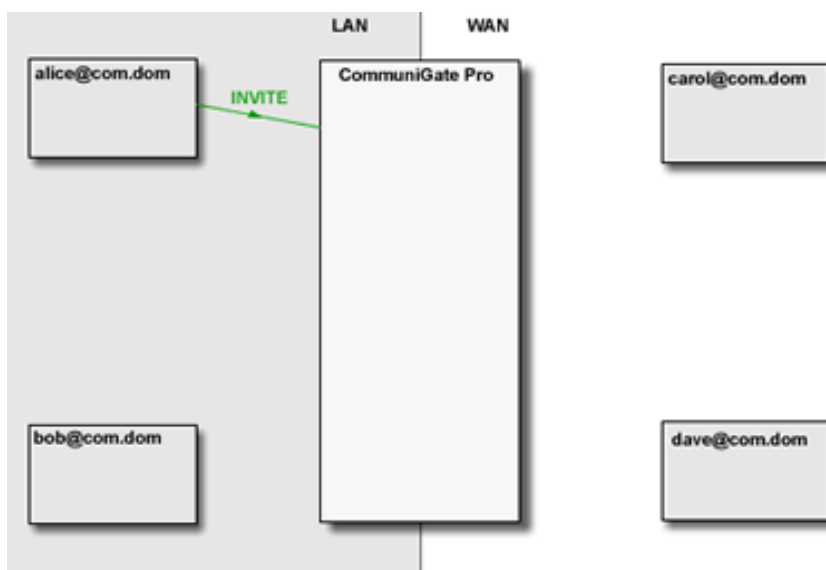
Near-End NAT Traversal

The CommuniGate Pro SIP Module detects 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:

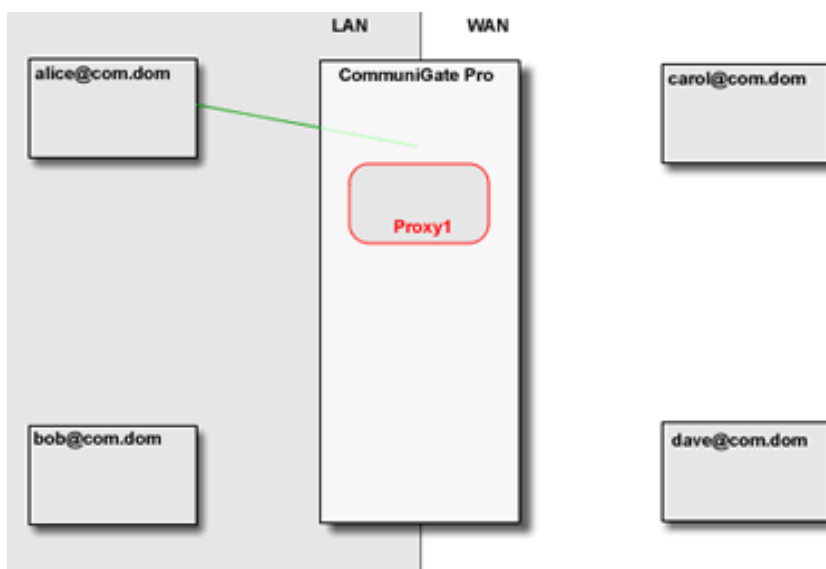
Step 1.



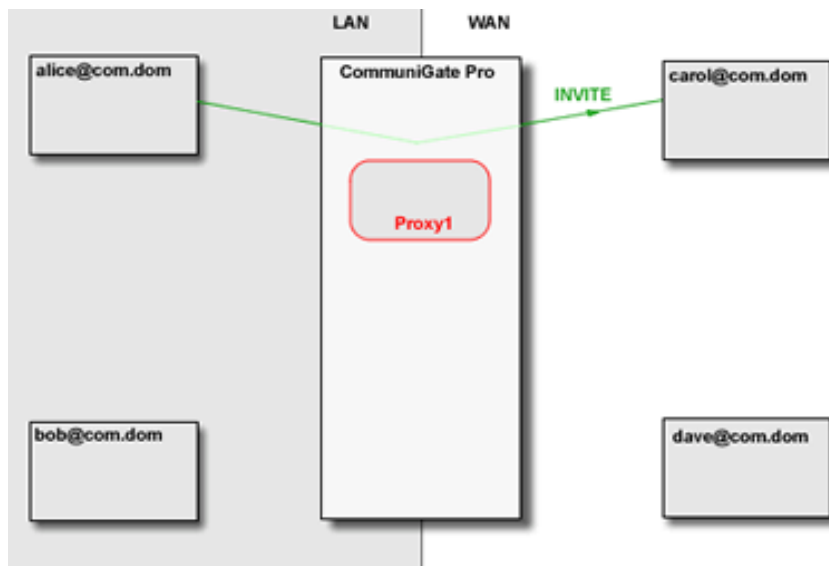
Step 2.



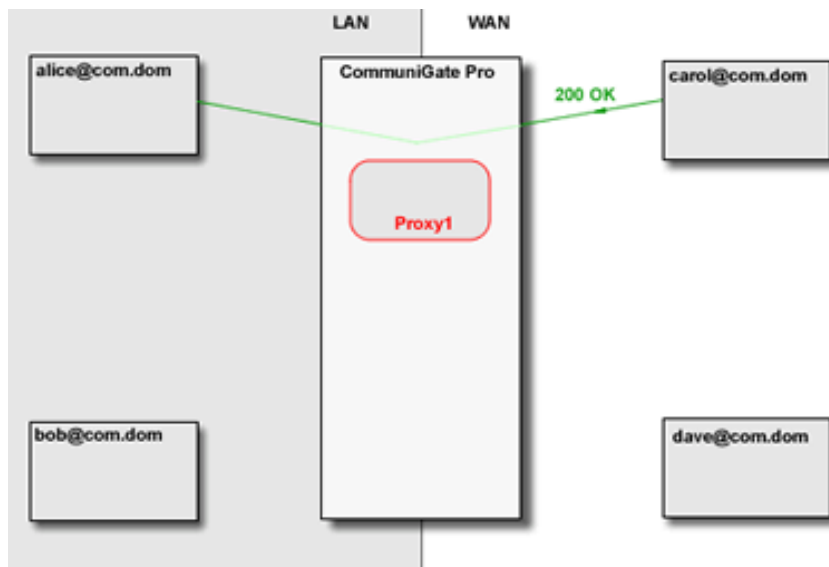
Step 3.



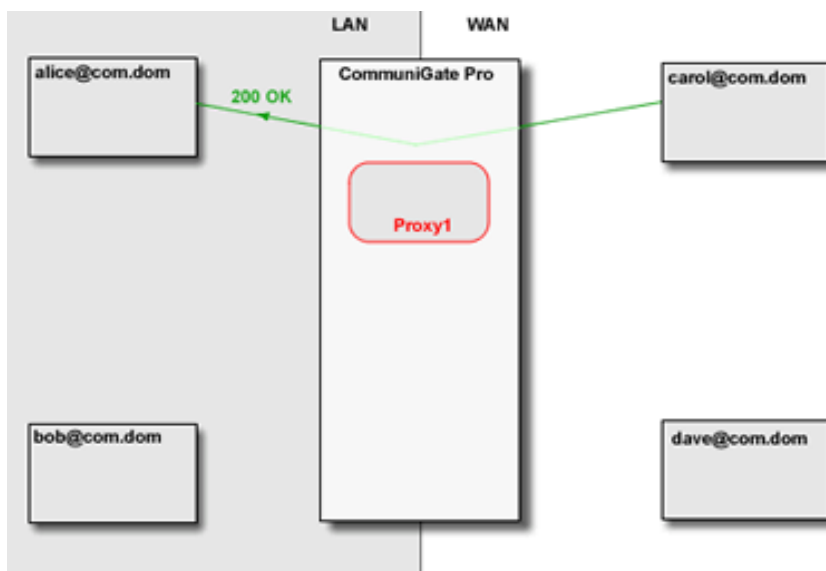
Step 4.



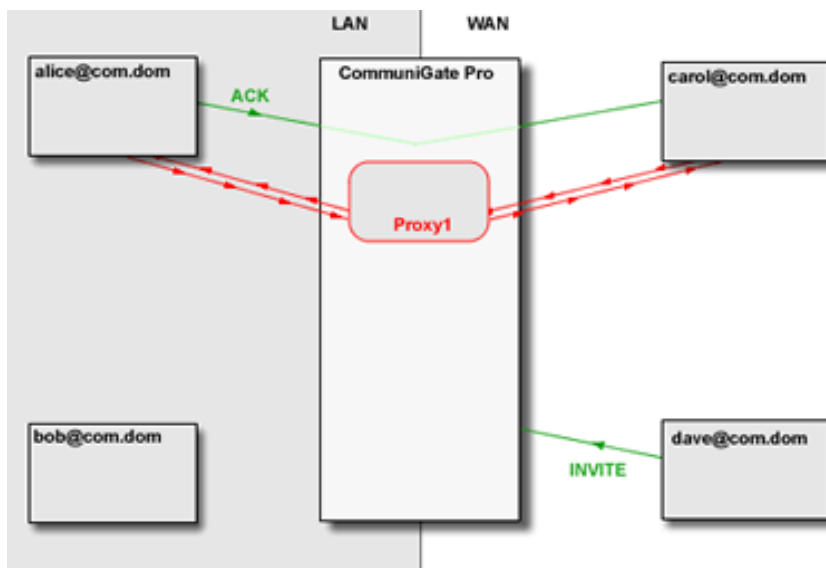
Step 5.



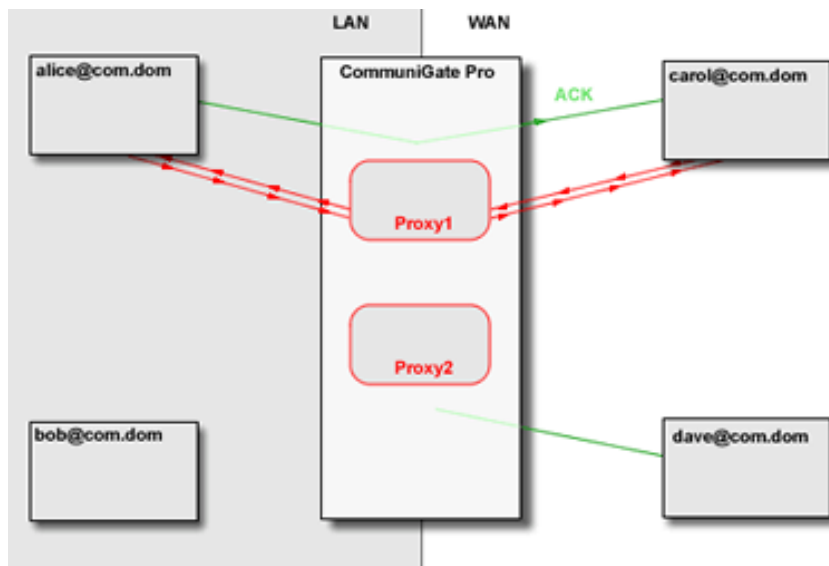
Step 6.



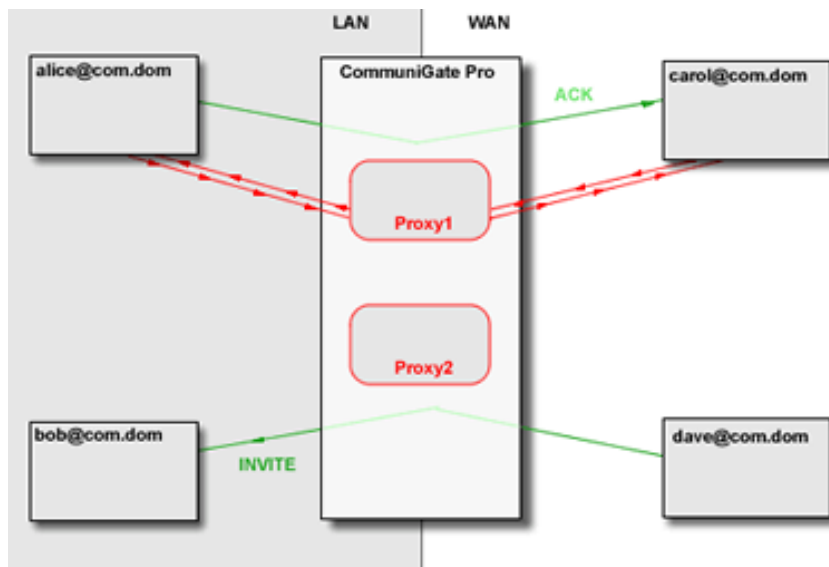
Step 7.



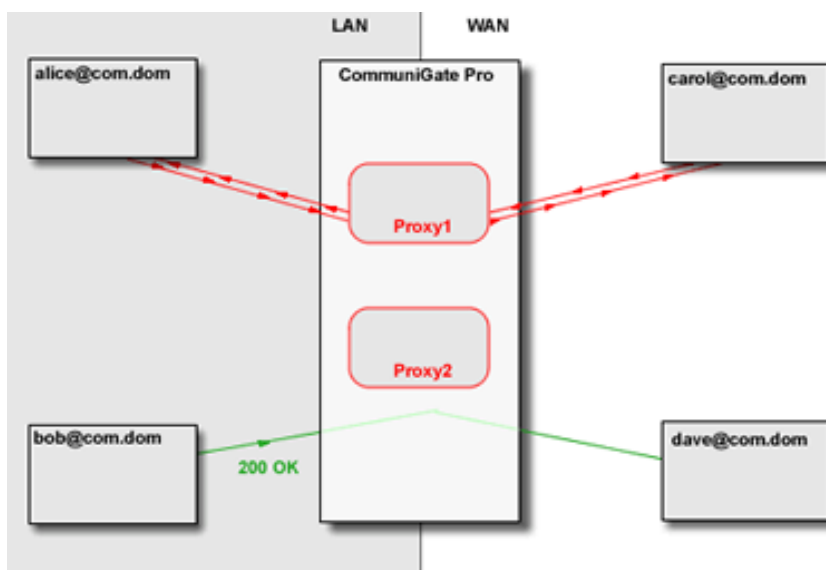
Step 8.



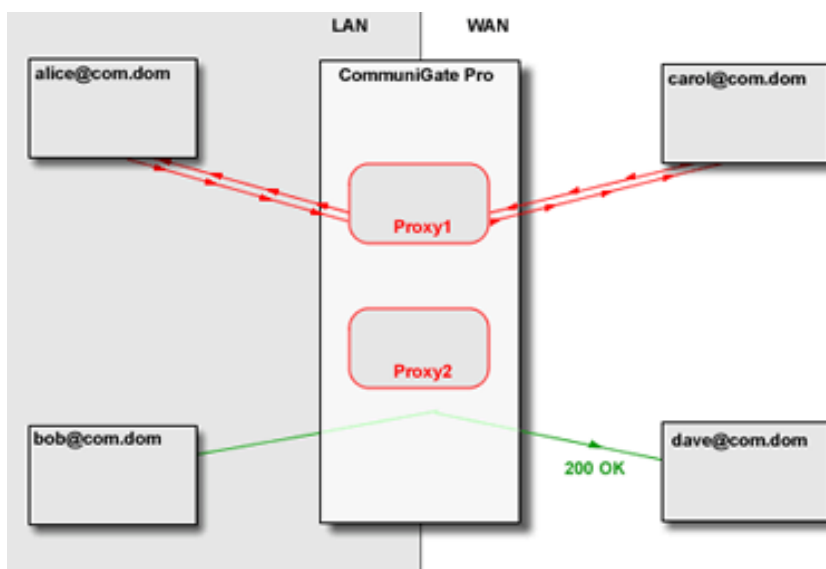
Step 9.



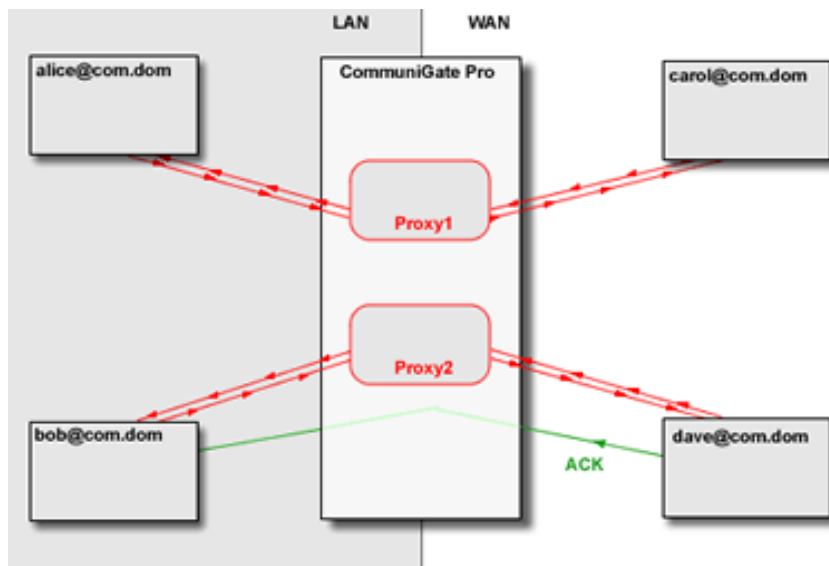
Step 10.



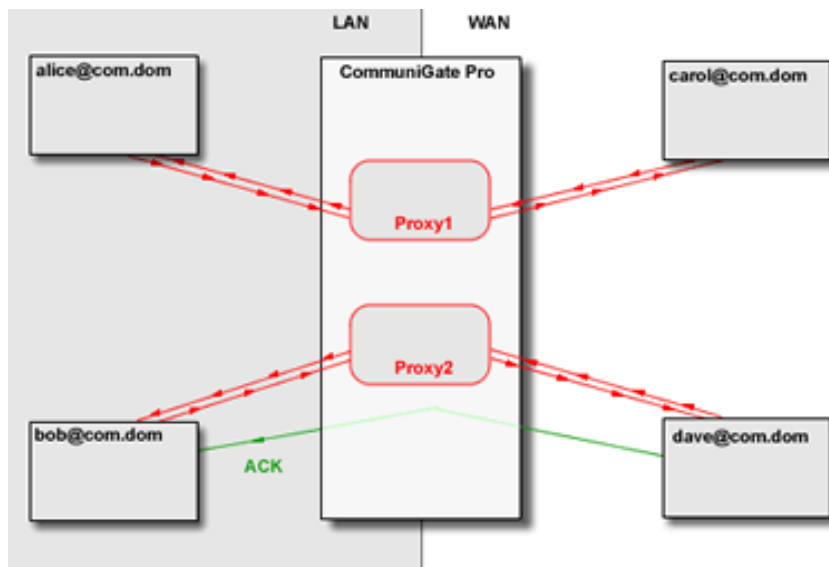
Step 11.



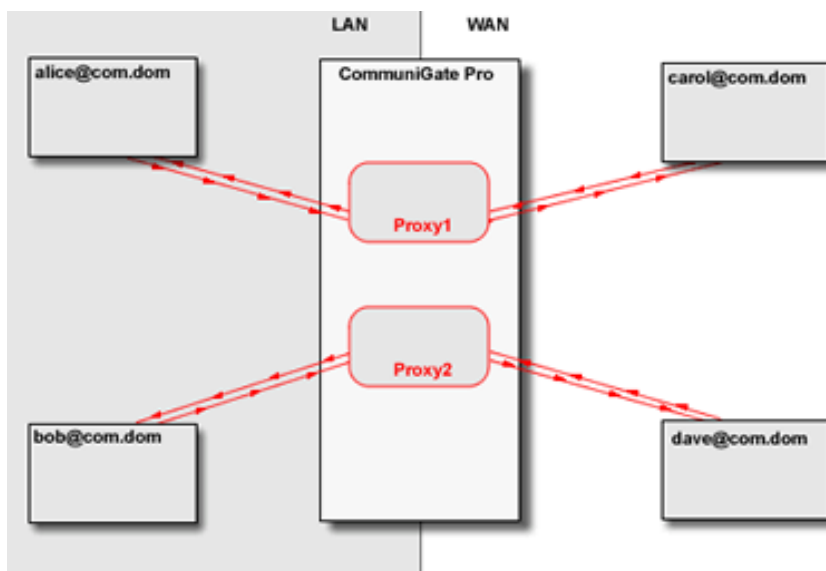
Step 12.



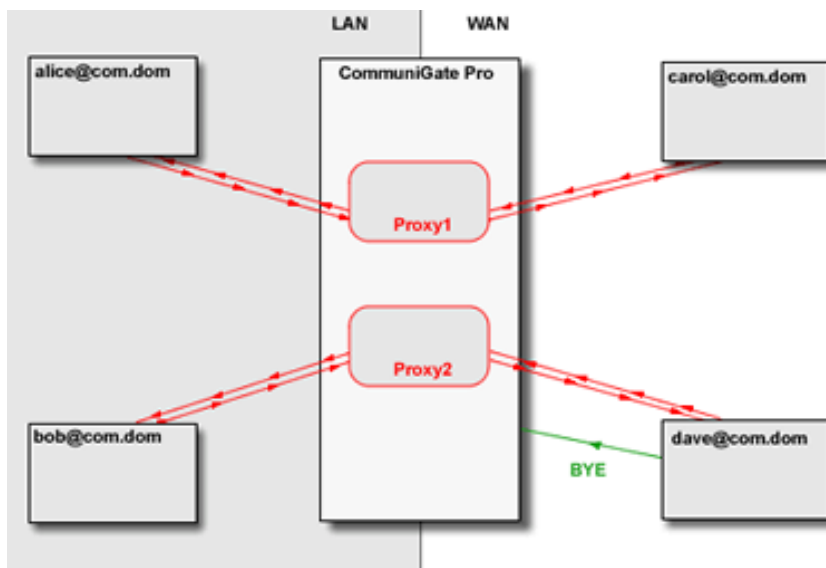
Step 13.



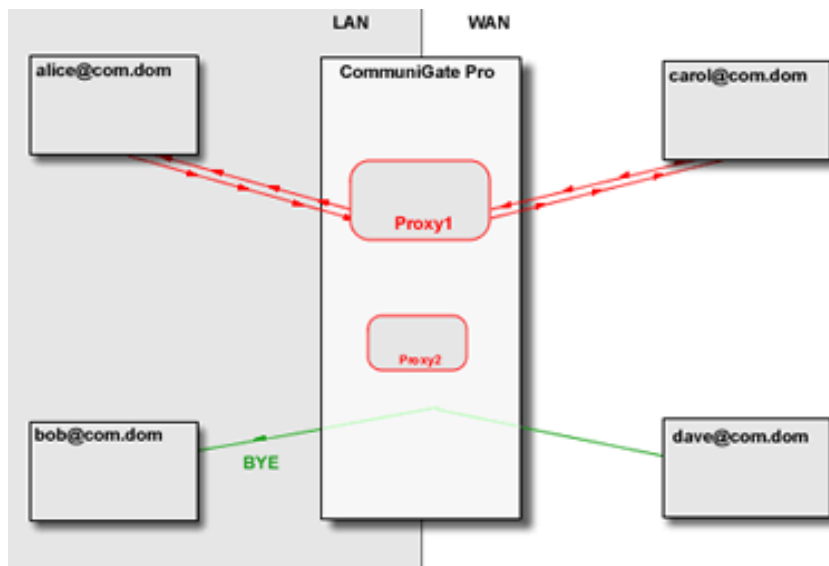
Step 14.



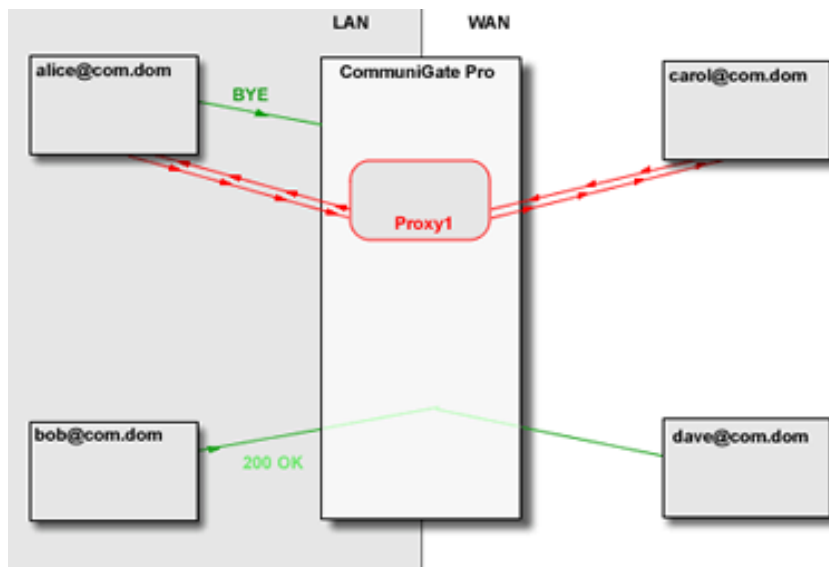
Step 15.



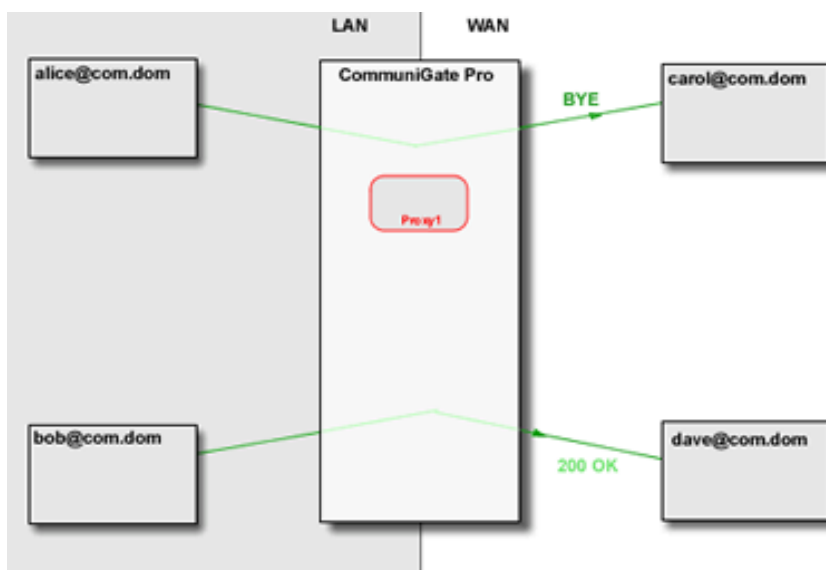
Step 16.



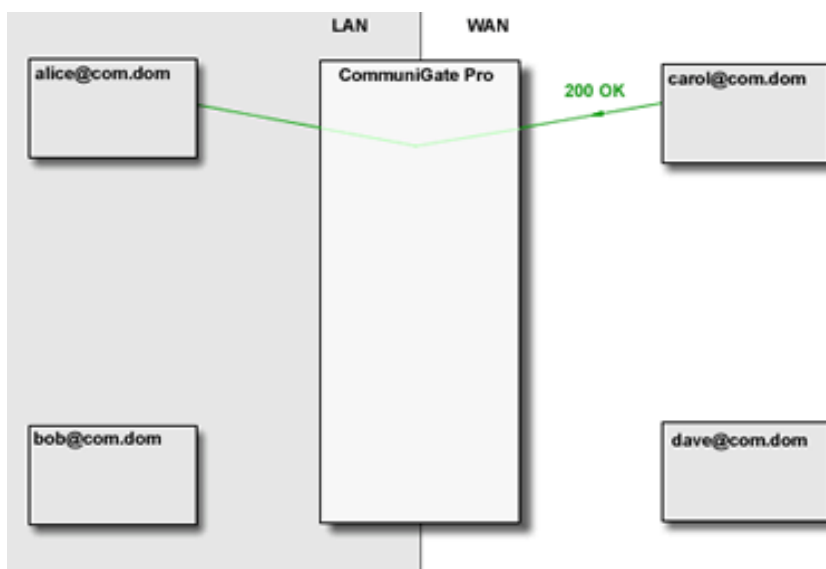
Step 17.



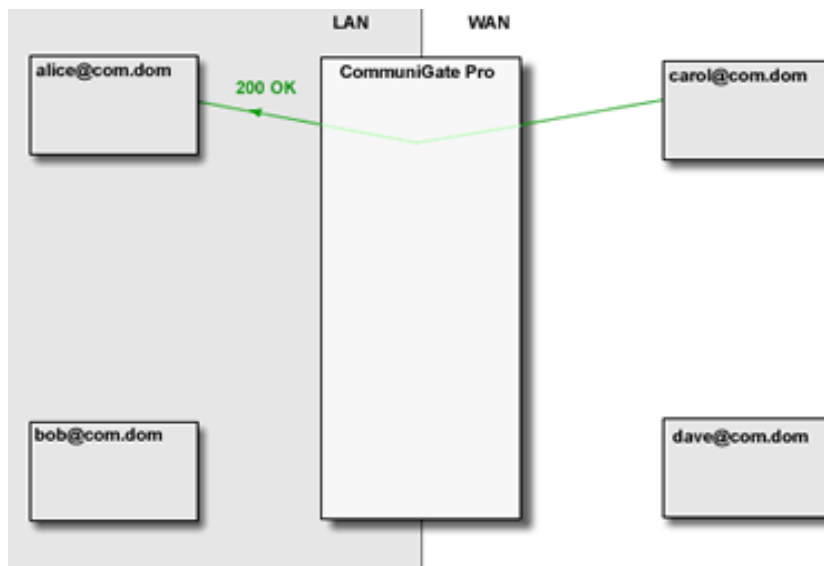
Step 18.



Step 19.



Step 20.



The CommuniGate Pro SIP Module detects session re-INVITE requests as well as BYE requests and update and removes the session 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

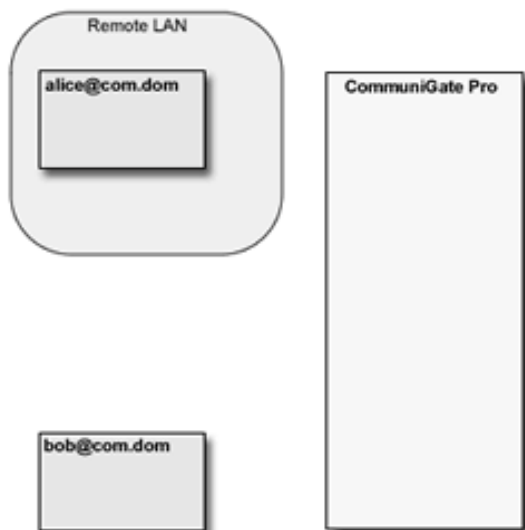
Note: If you need the Media Stream Proxy functionality, make sure that the LAN and NAT data is specified correctly on the [LAN IPs](#) settings page.

Far-End NAT Traversal

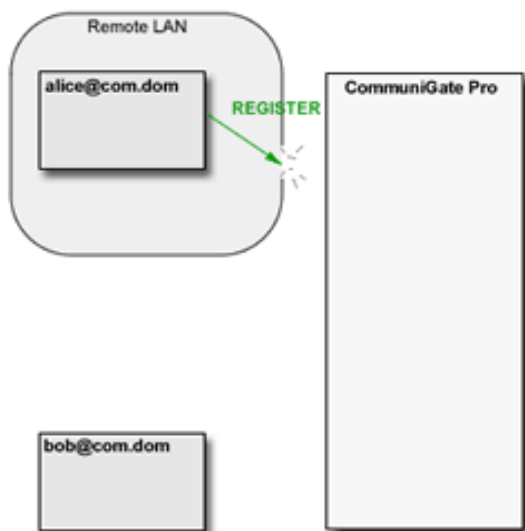
The CommuniGate Pro SIP Module also provides the "far-end" NAT traversal capabilities by detecting requests coming from clients located behind remote Firewall/NATs.

The Module adds appropriate Record-Route and Path headers to these requests and it builds media proxies to relay traffic to and from those clients.

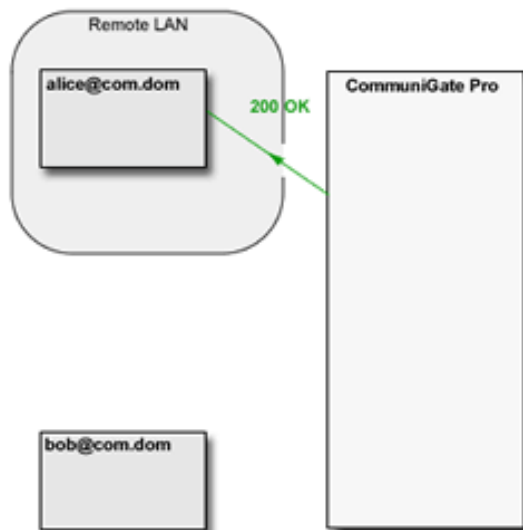
Step 1.



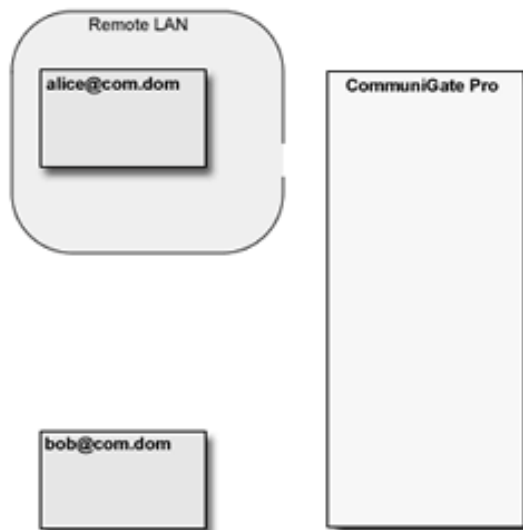
Step 2.



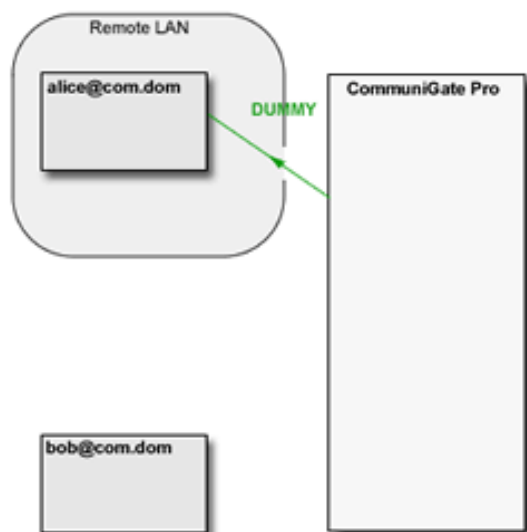
Step 3.



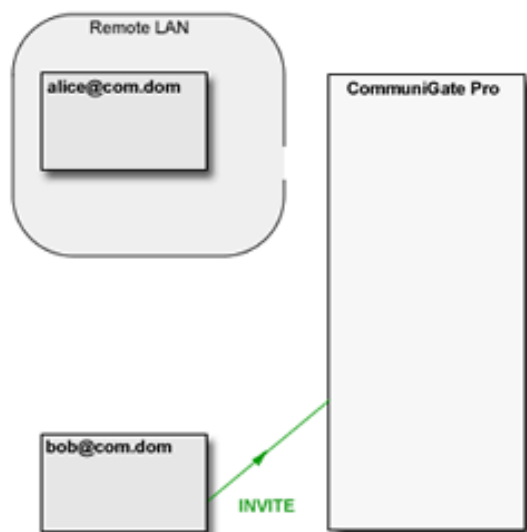
Step 4.



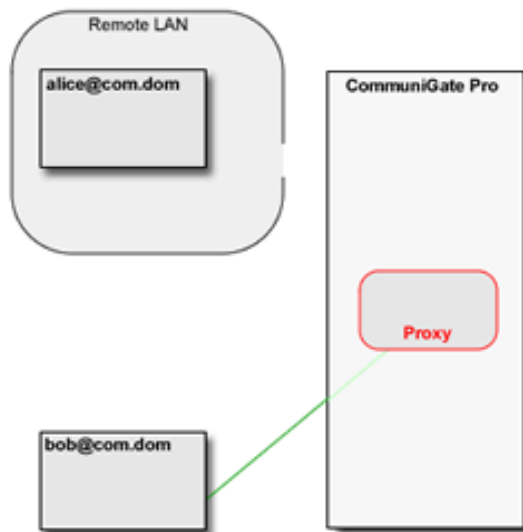
Step 5.



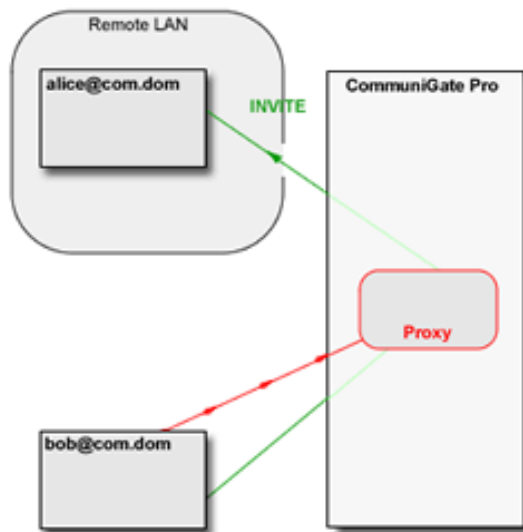
Step 6.



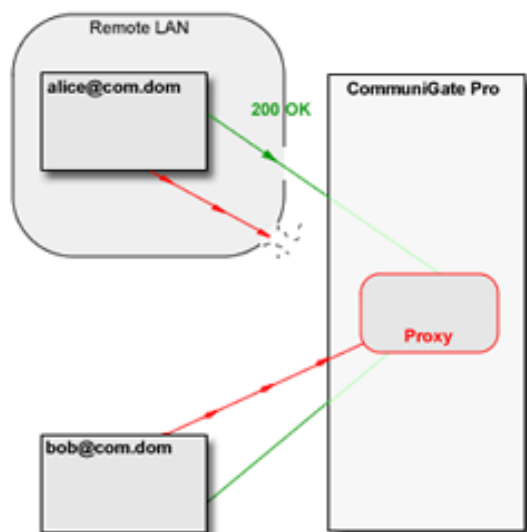
Step 7.



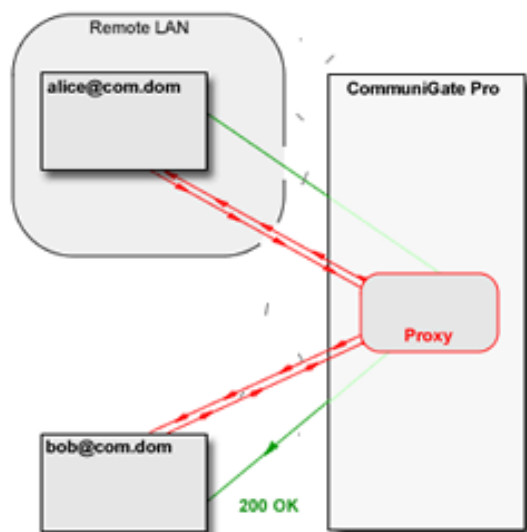
Step 8.



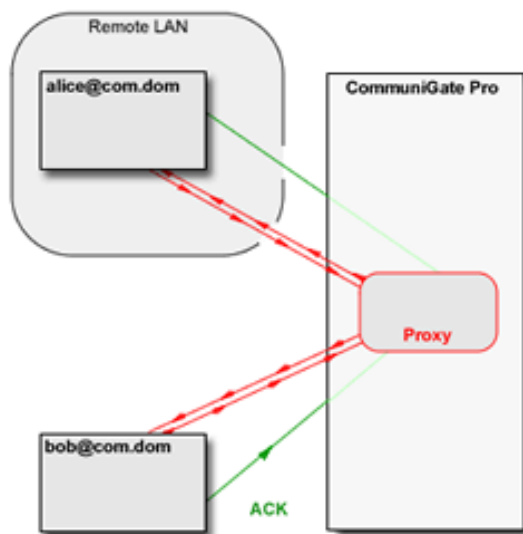
Step 9.



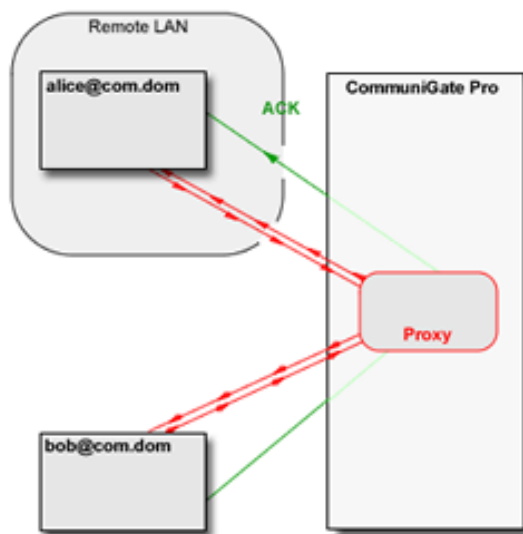
Step 10.



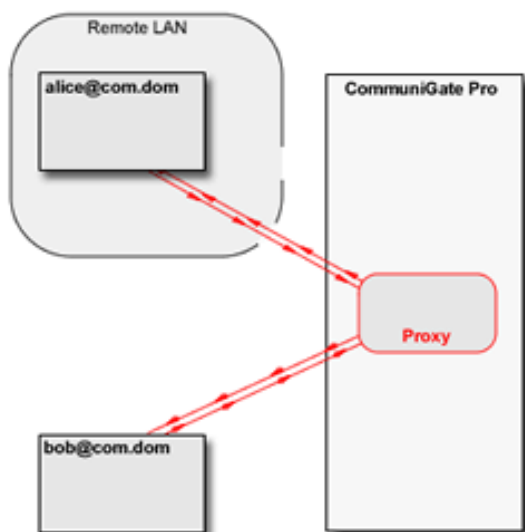
Step 11.



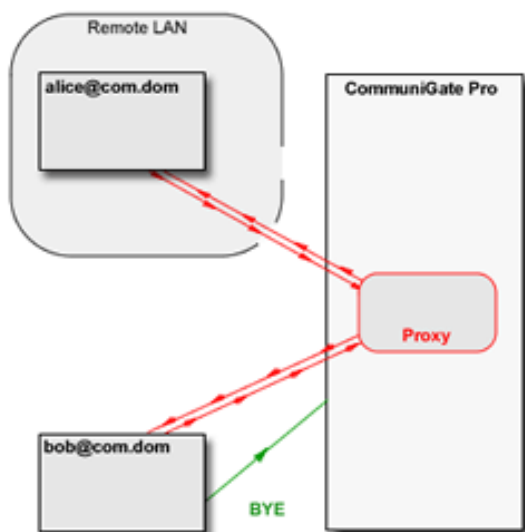
Step 12.



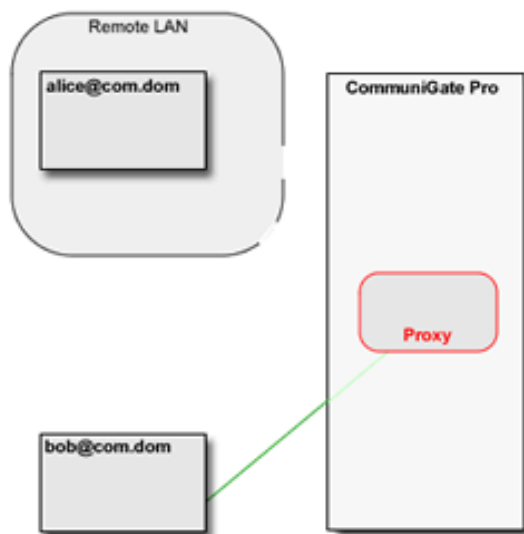
Step 13.



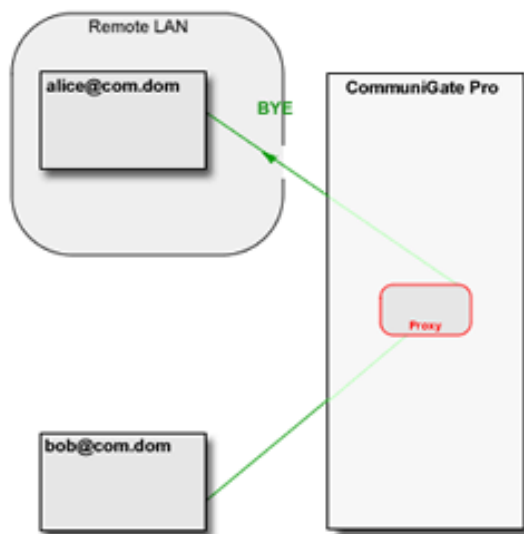
Step 14.



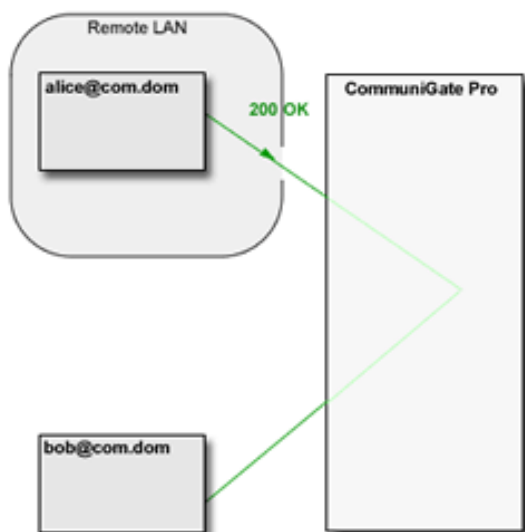
Step 15.



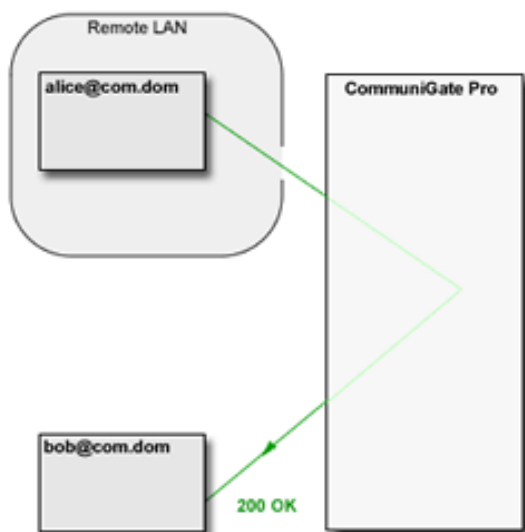
Step 16.



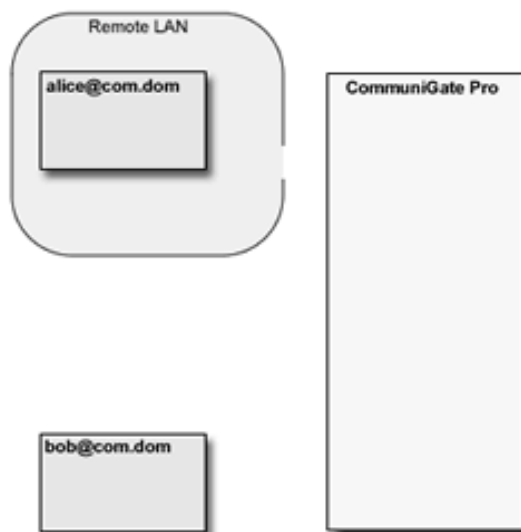
Step 17.



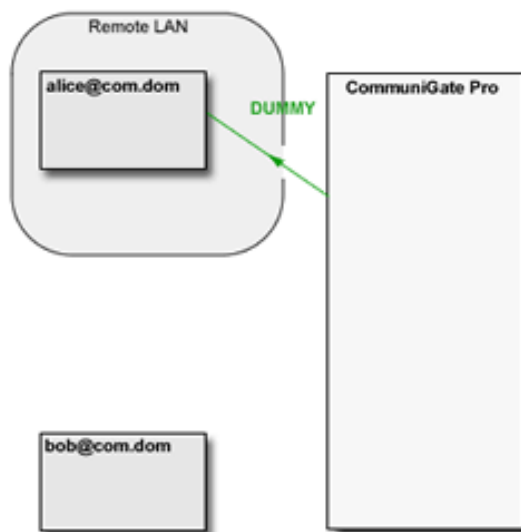
Step 18.



Step 19.



Step 20.



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.

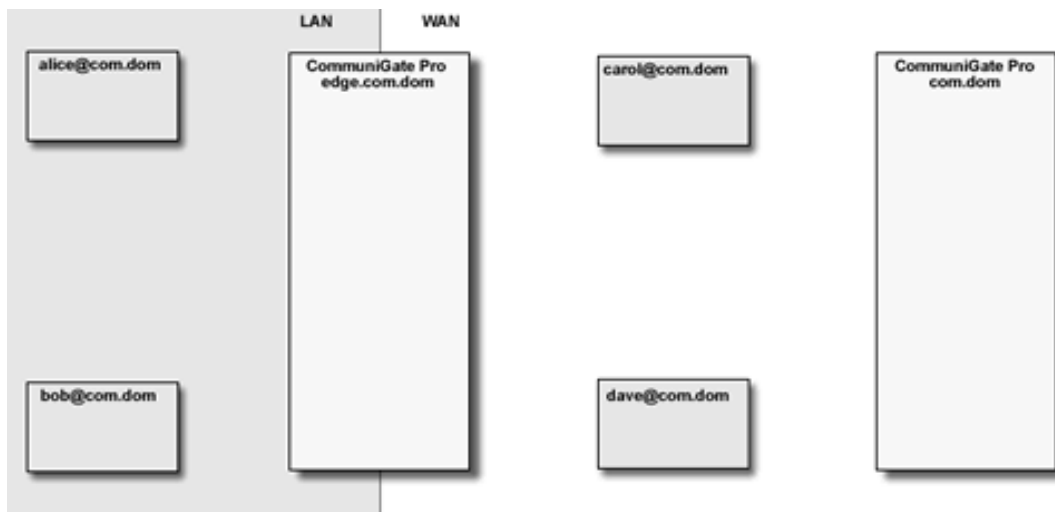
Note: due to the nature of the TCP protocol and the Firewall concept, it is not possible (in general) to open a TCP connection to a client behind a far-end NAT ("near-end" NAT configurations do not have this problem). This means that clients behind a far-end NAT cannot initiate TCP (T.120) sessions. 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.

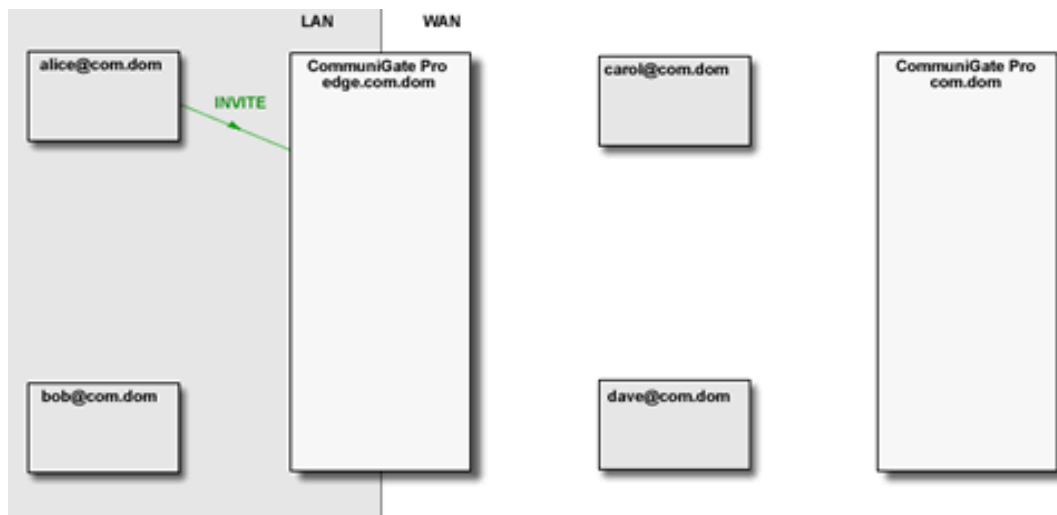
Edge Services

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.

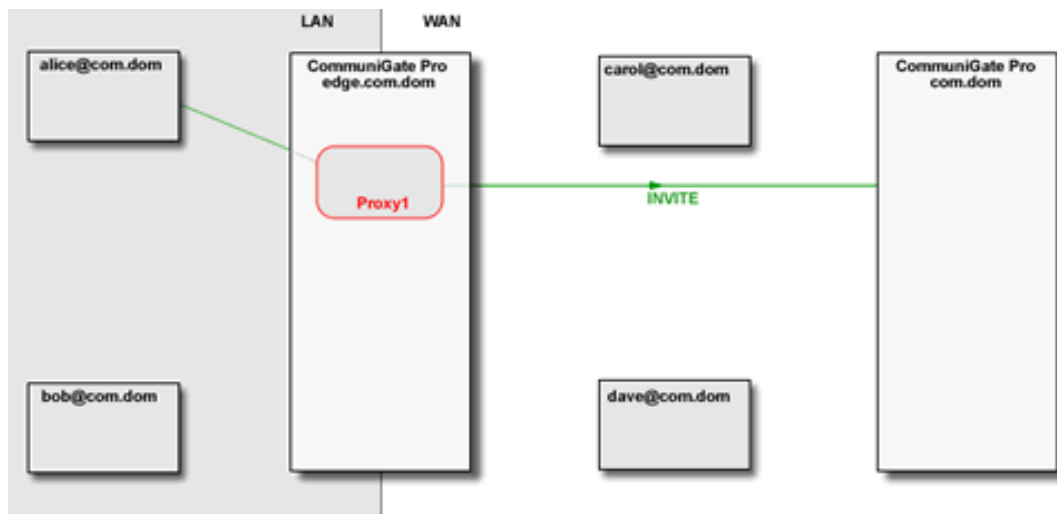
Step 1.



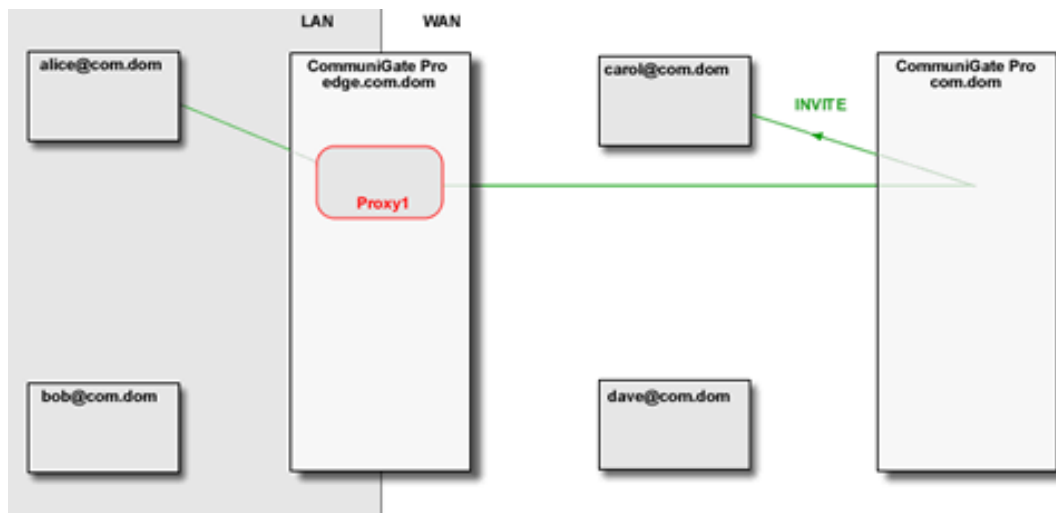
Step 2.



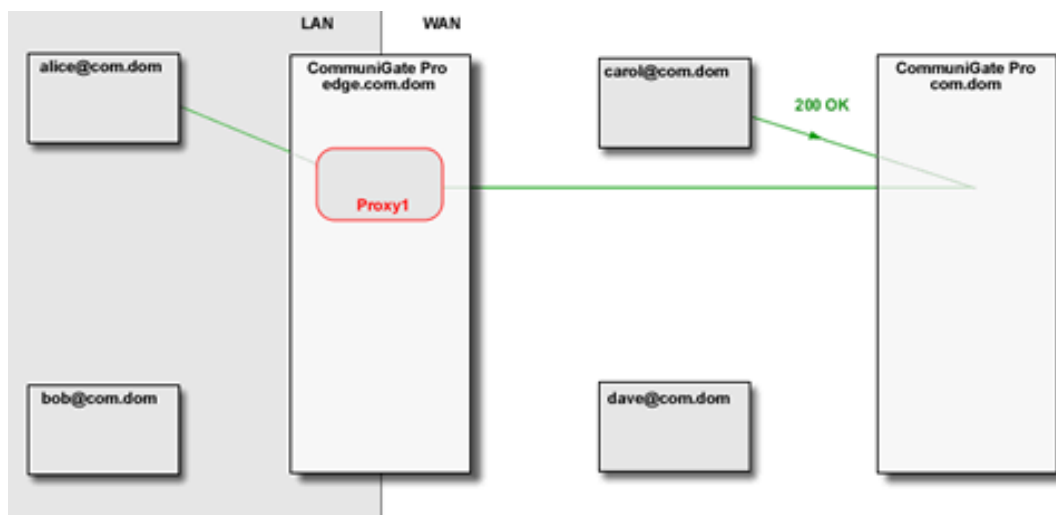
Step 3.



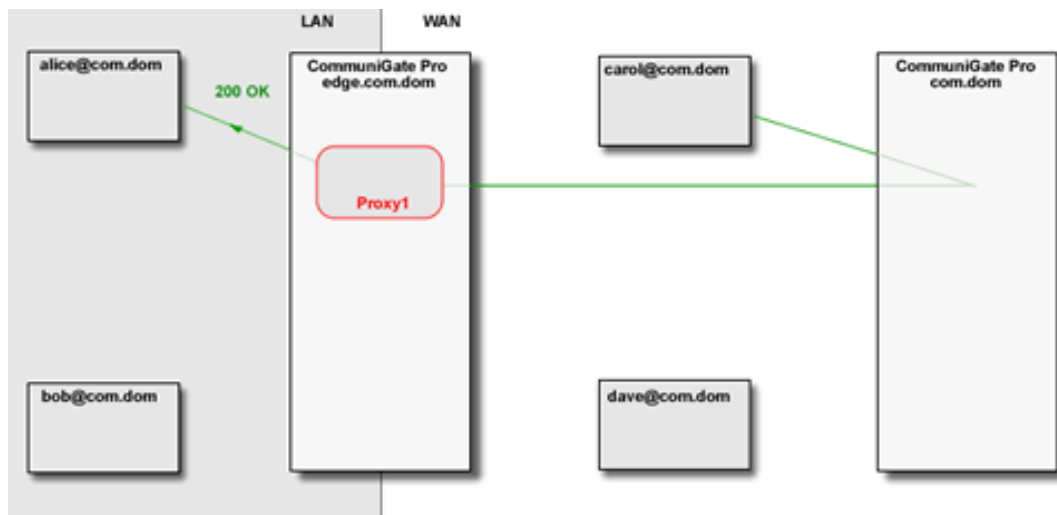
Step 4.



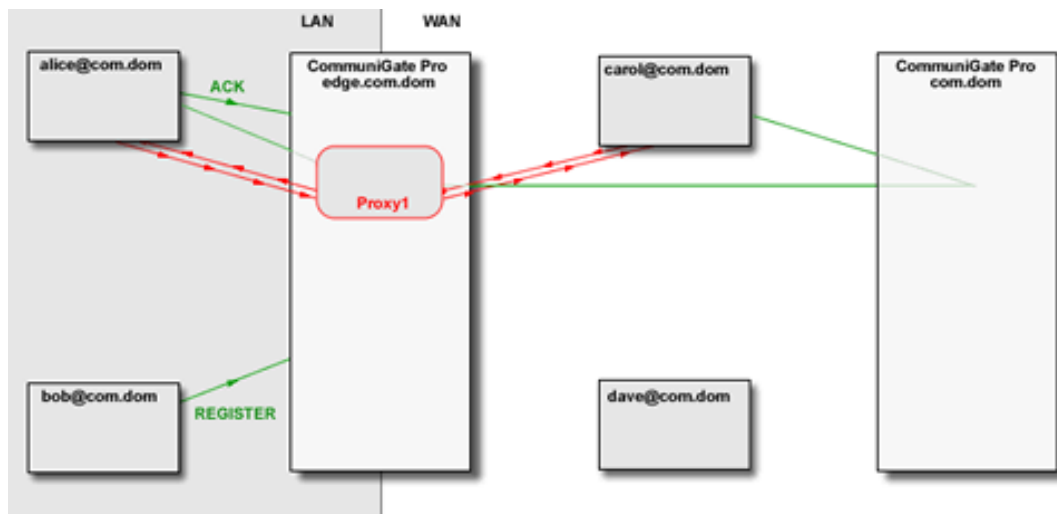
Step 5.



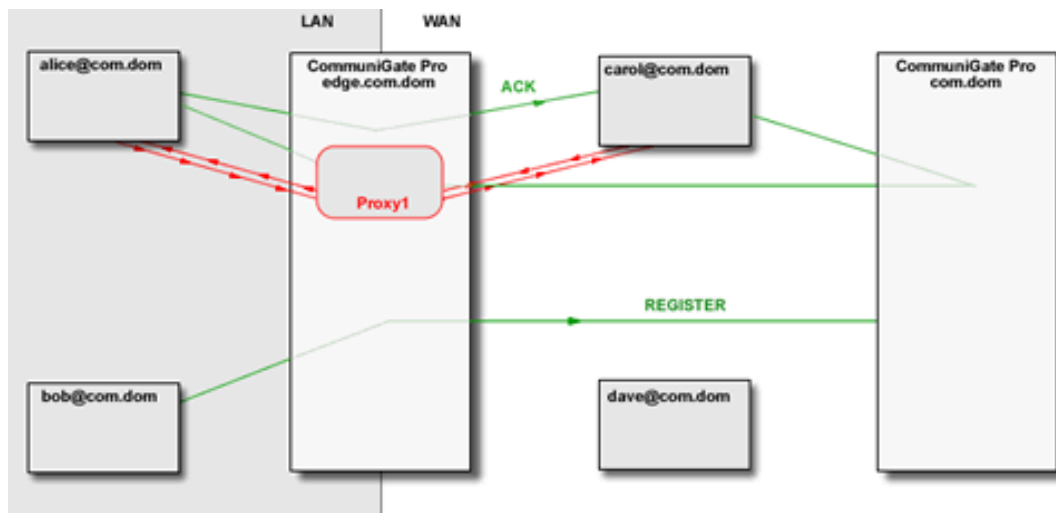
Step 6.



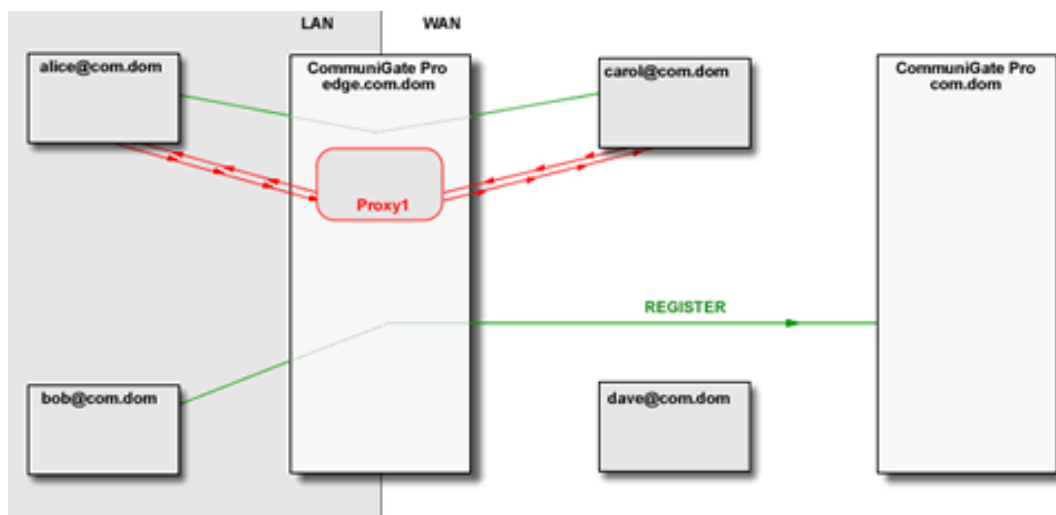
Step 7.



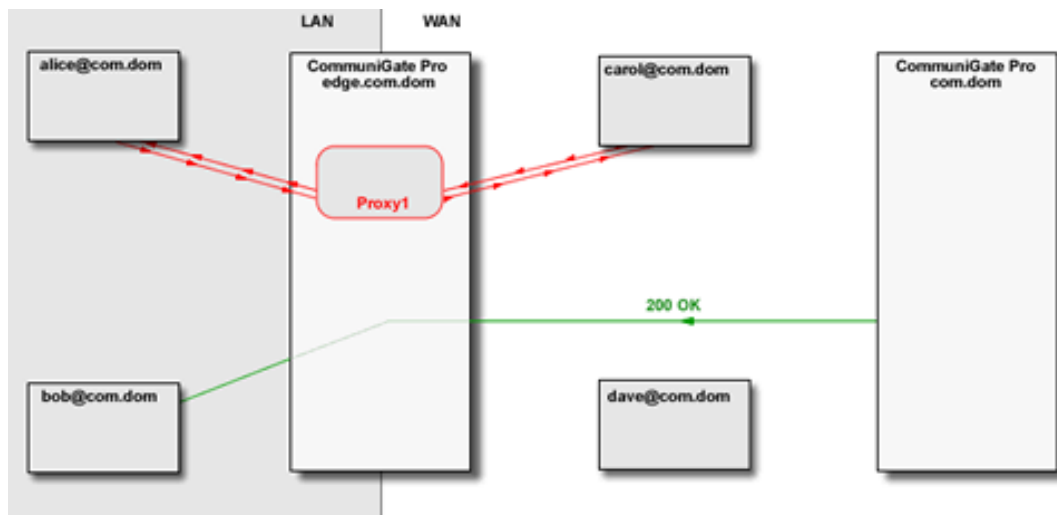
Step 8.



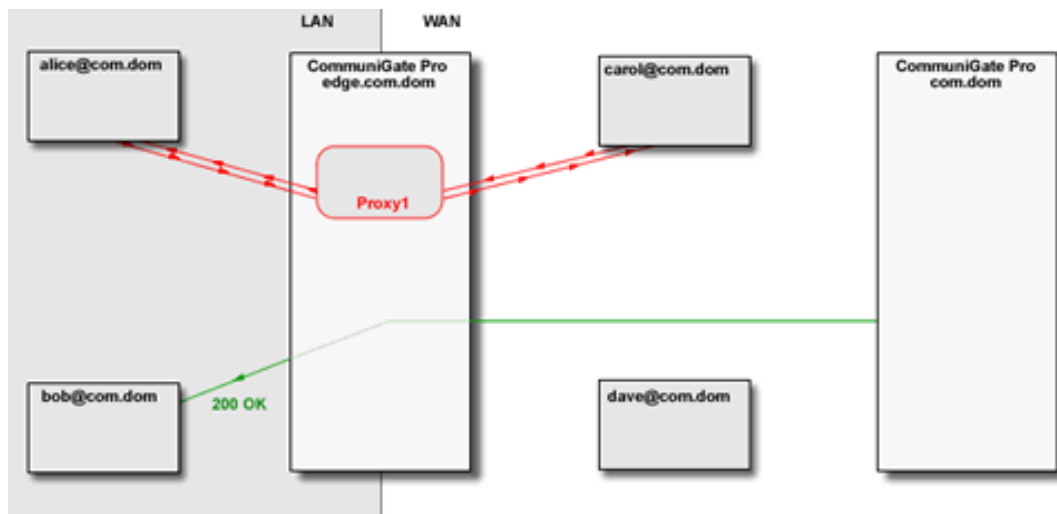
Step 9.



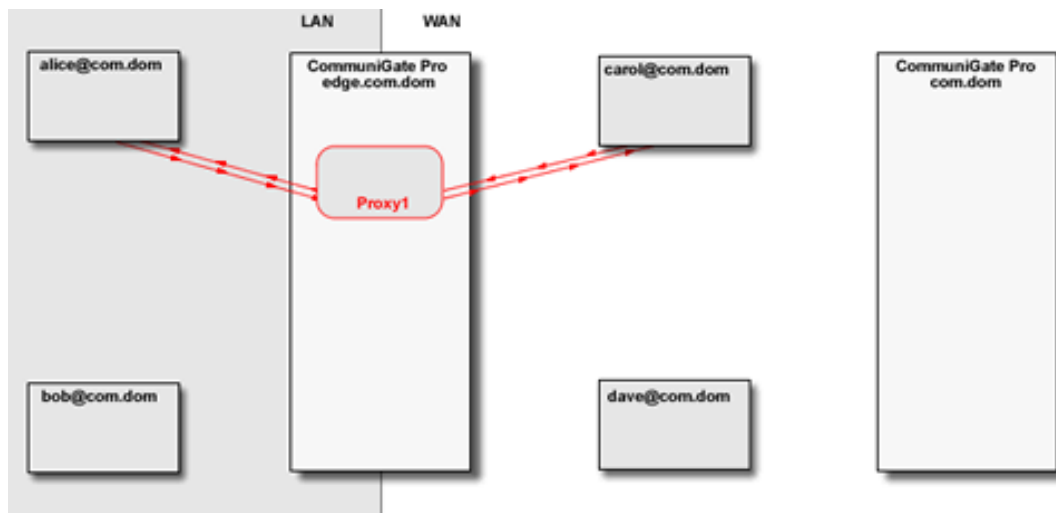
Step 10.



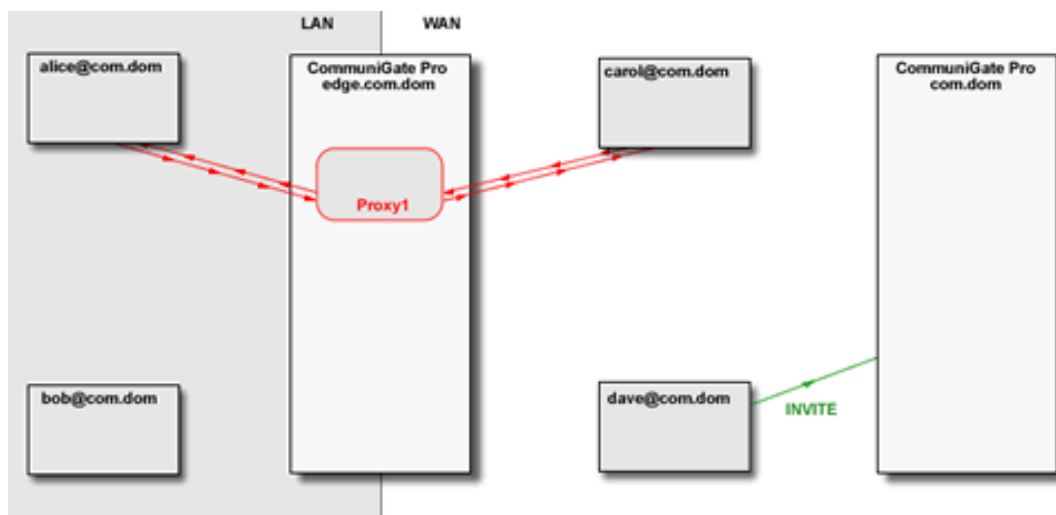
Step 11.



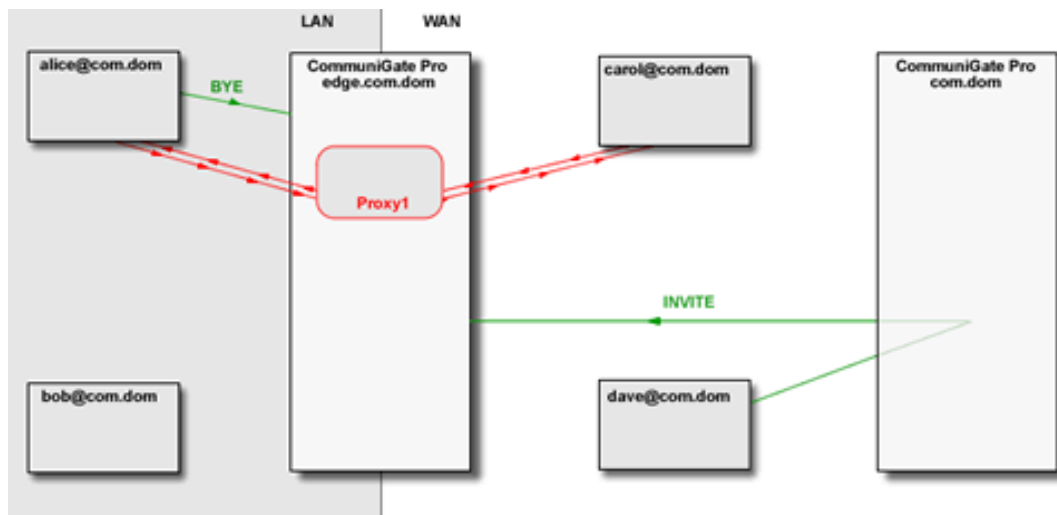
Step 12.



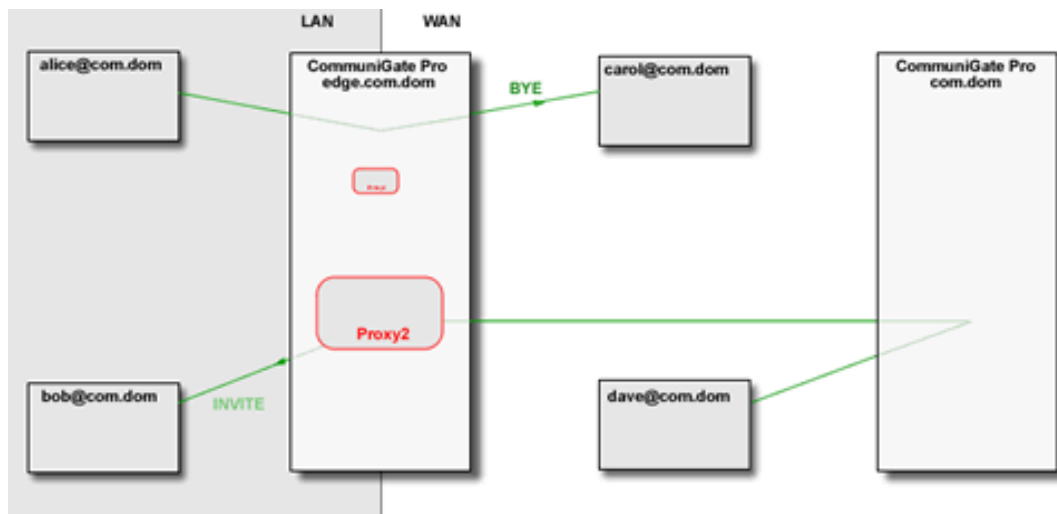
Step 13.



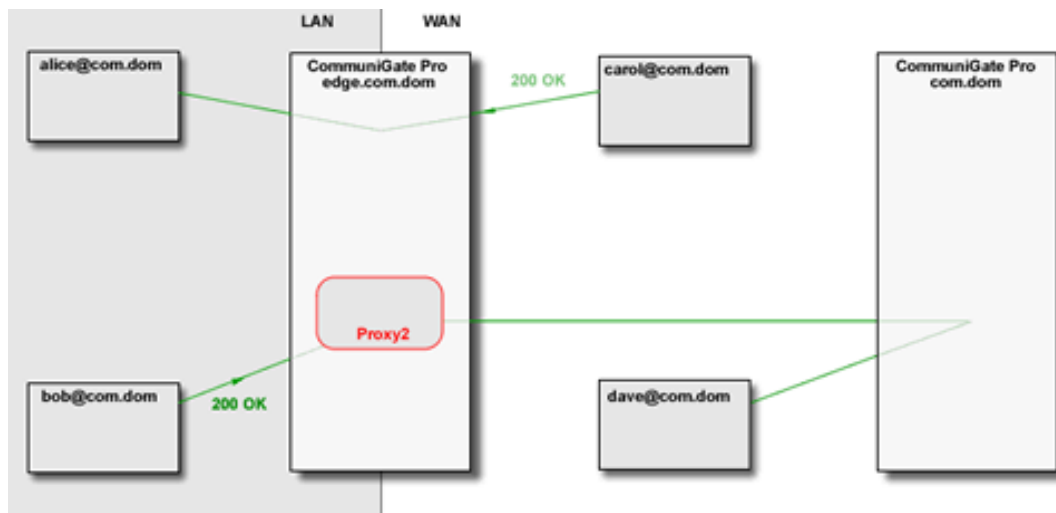
Step 14.



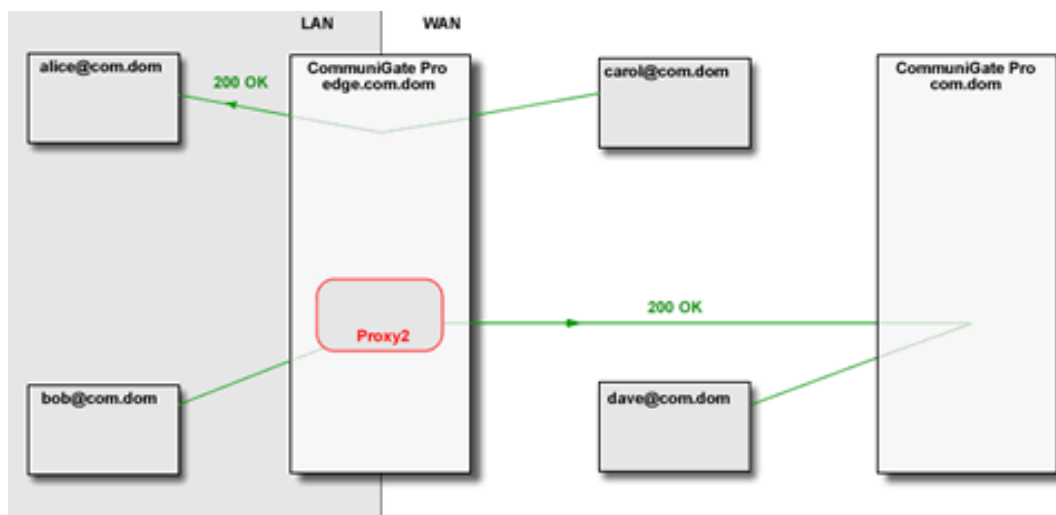
Step 15.



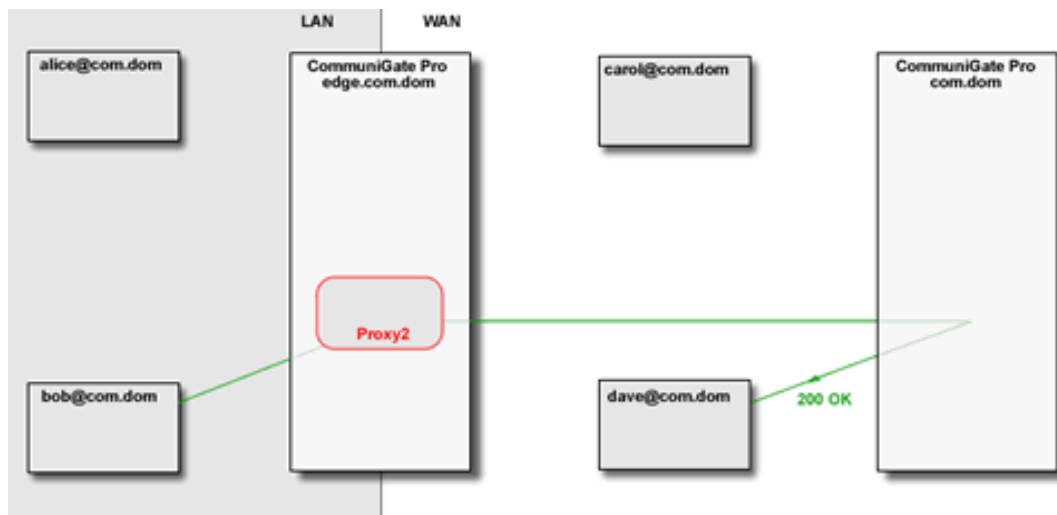
Step 16.



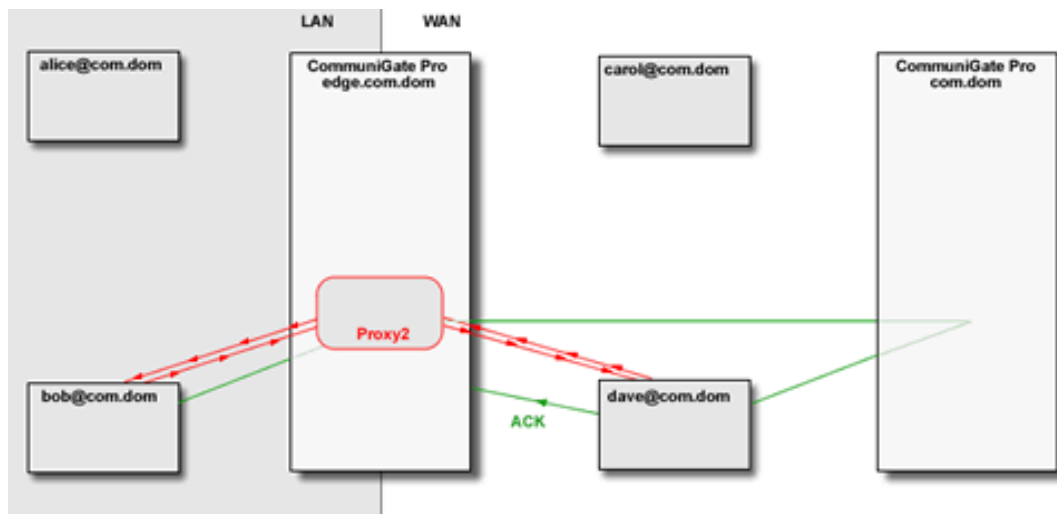
Step 17.



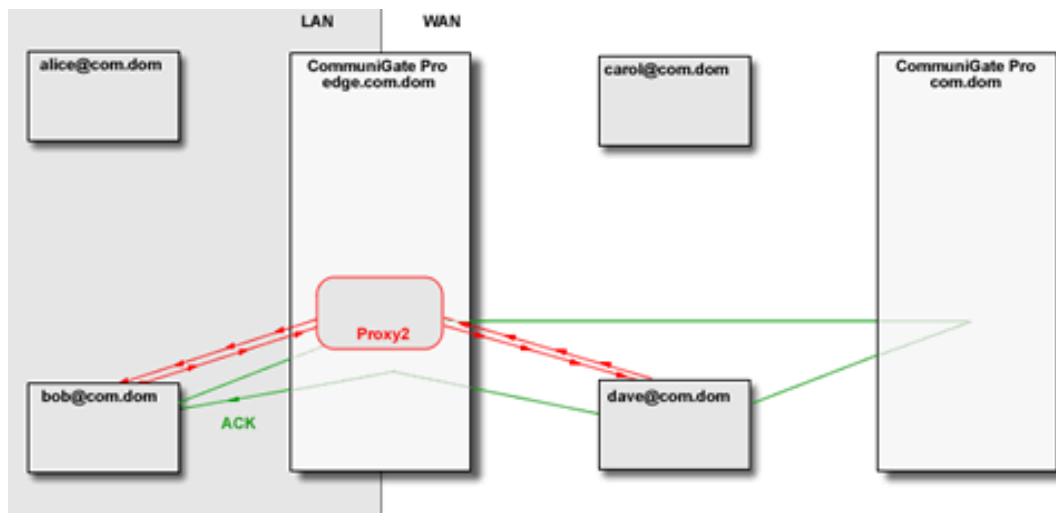
Step 18.



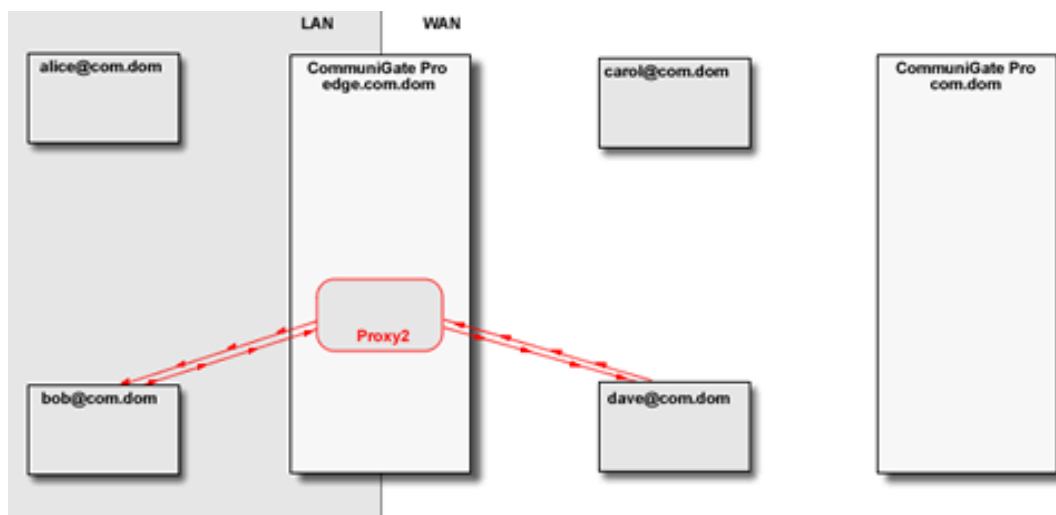
Step 19.



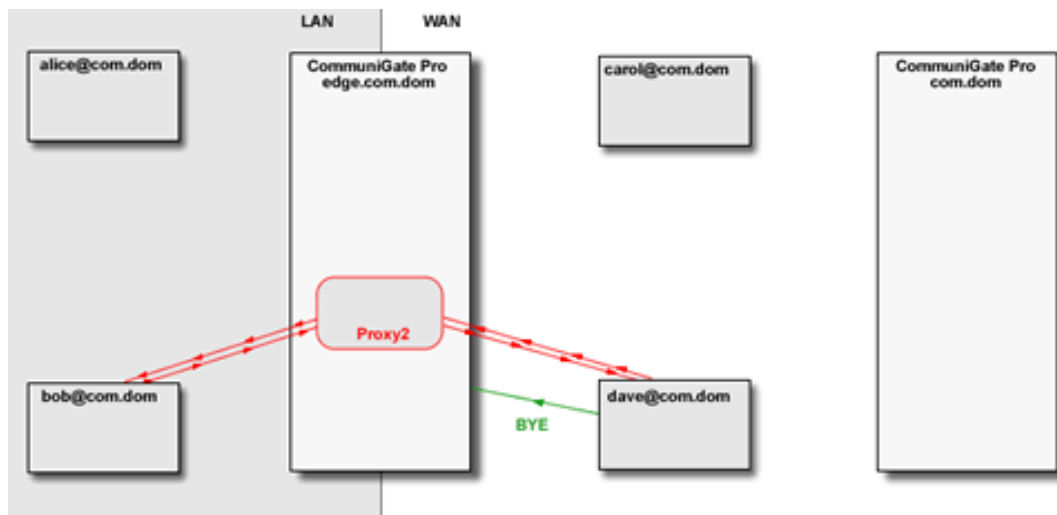
Step 20.



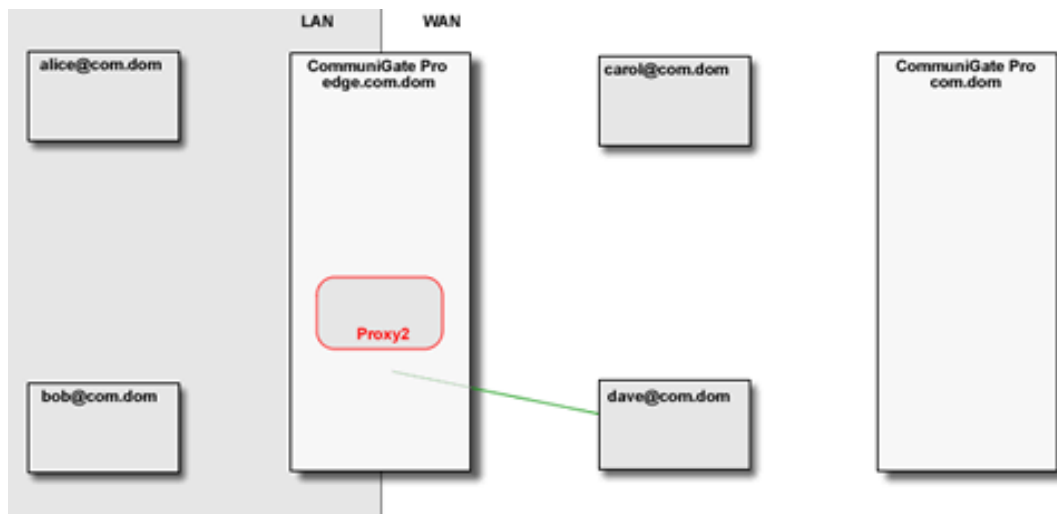
Step 21.



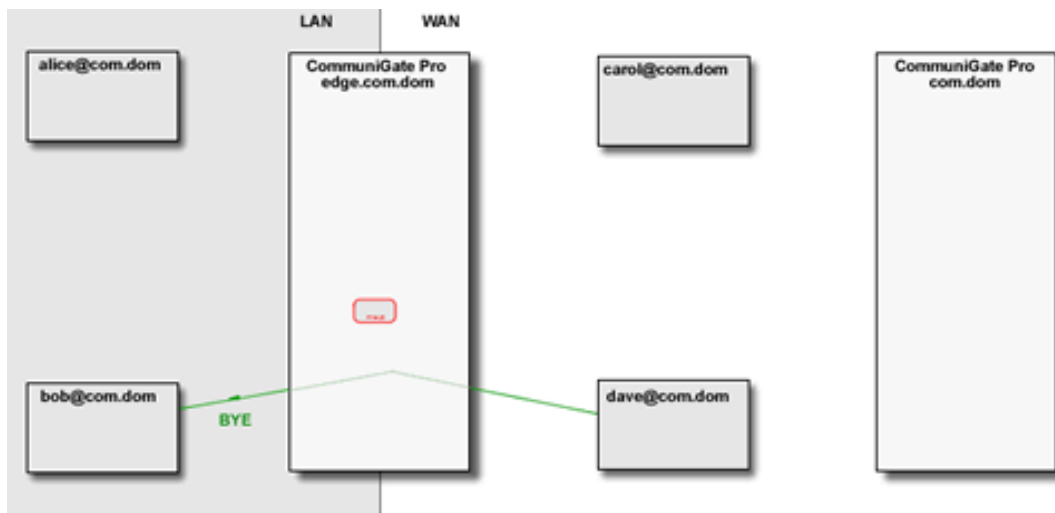
Step 22.



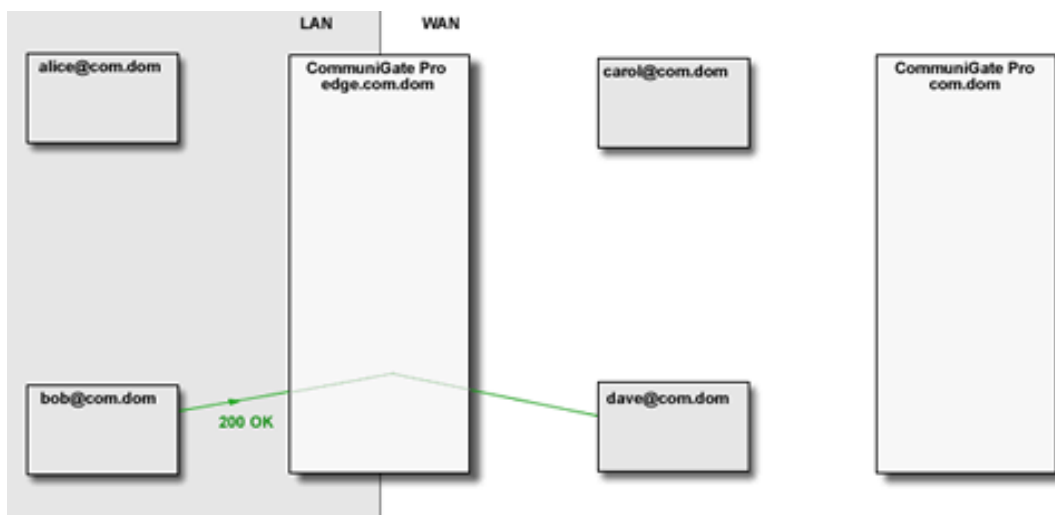
Step 23.



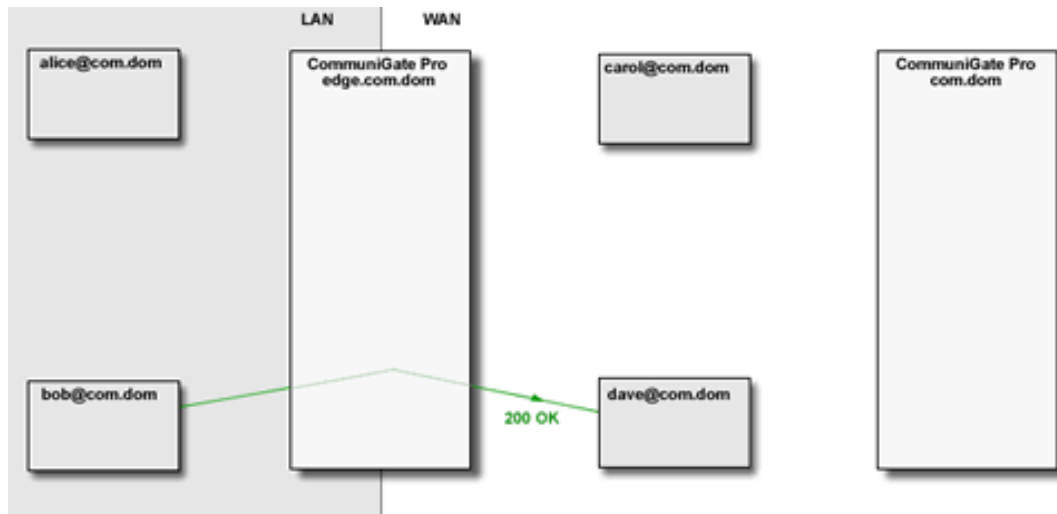
Step 24.



Step 25.

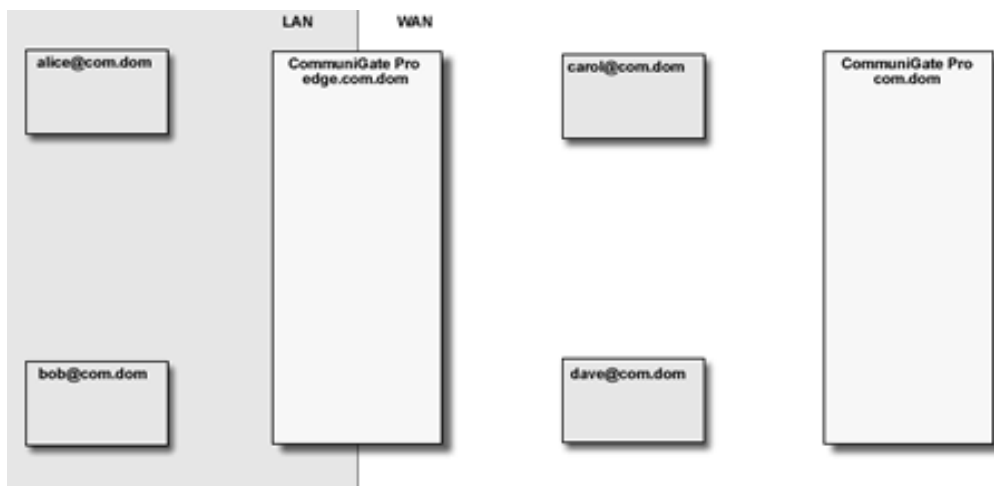


Step 26.

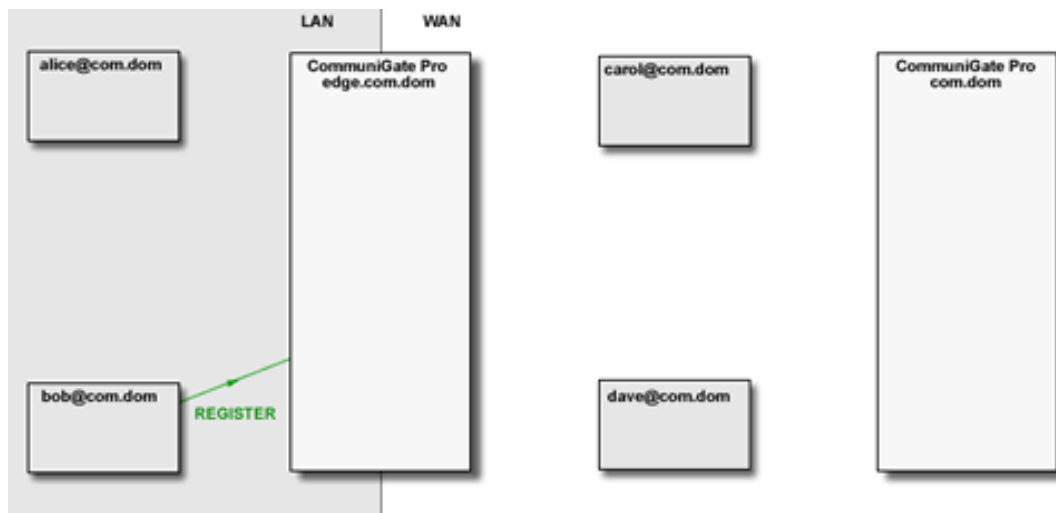


The CommuniGate Pro SIP Module detects "media loops", when a call placed from within LAN is proxied to WAN, and then proxied back to the same LAN. In this case the Media Proxies are removed, eliminating unnecessary overhead, and allowing SIP clients to communicate directly within one LAN, while proving registrar services outside that LAN.

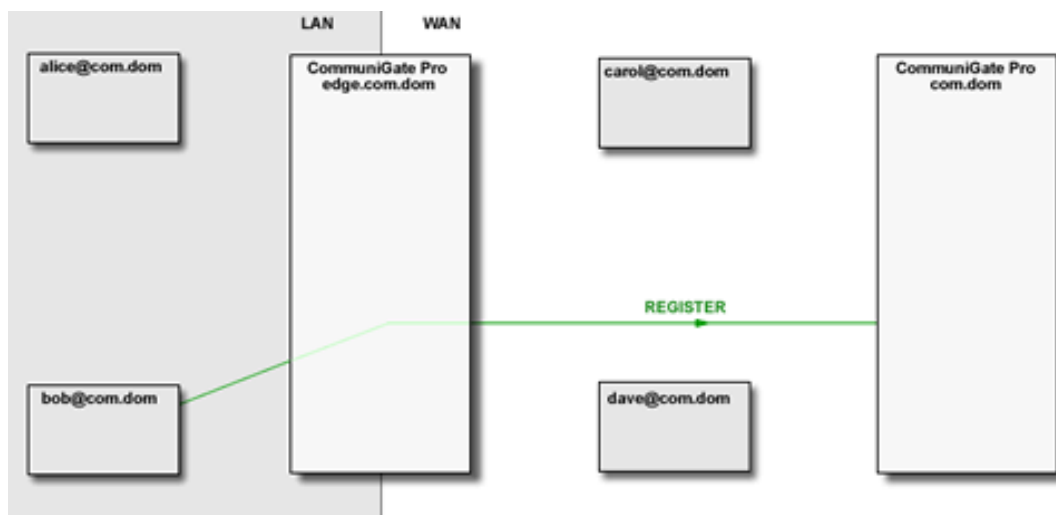
Step 1



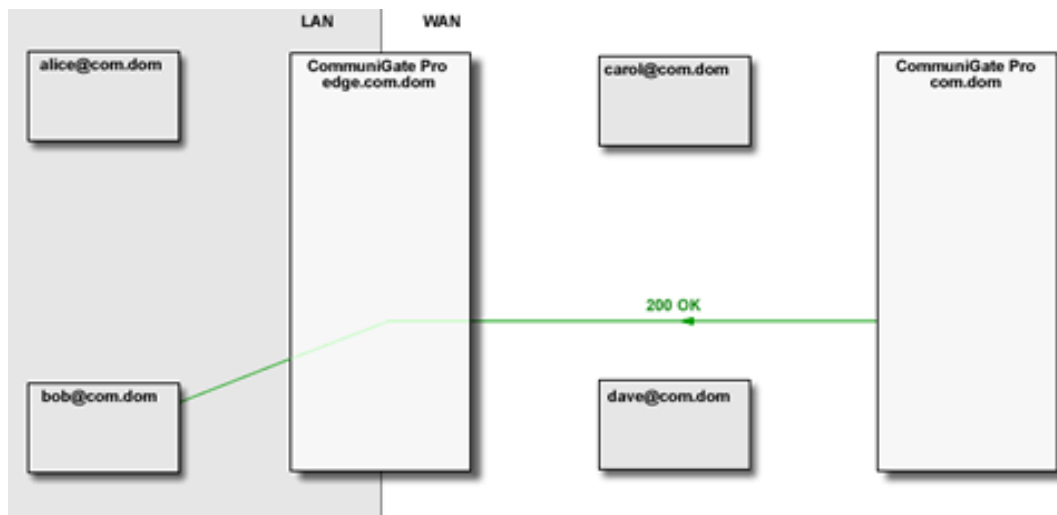
Step 2.



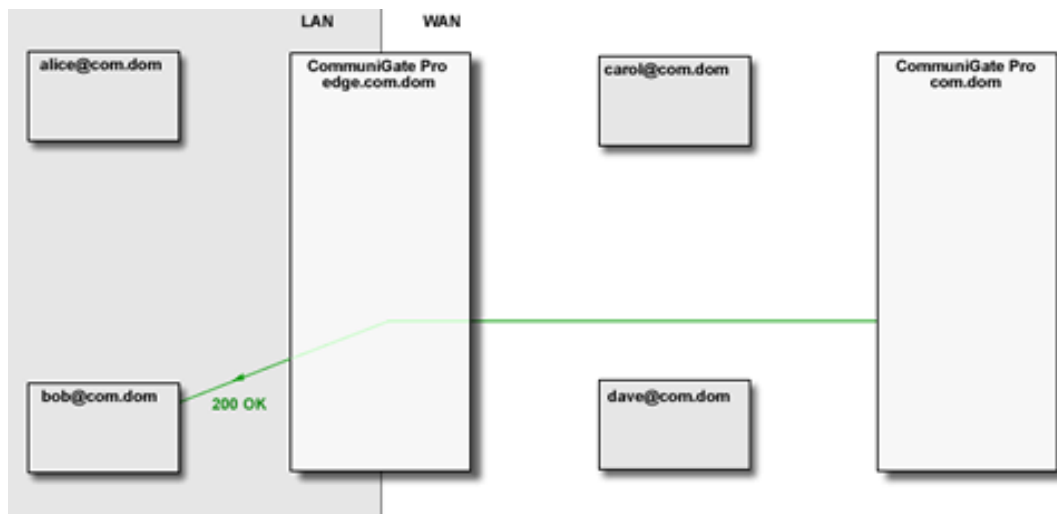
Step 3.



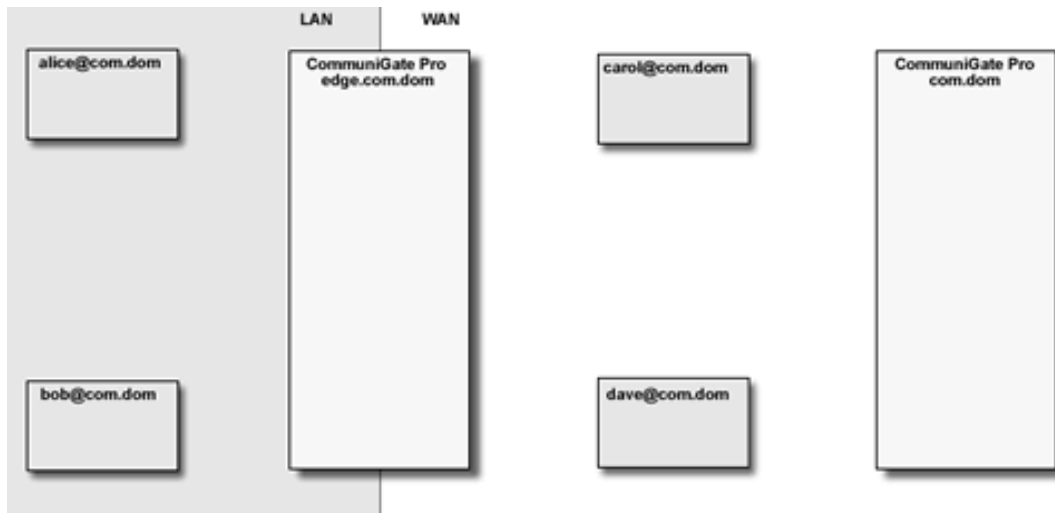
Step 4.



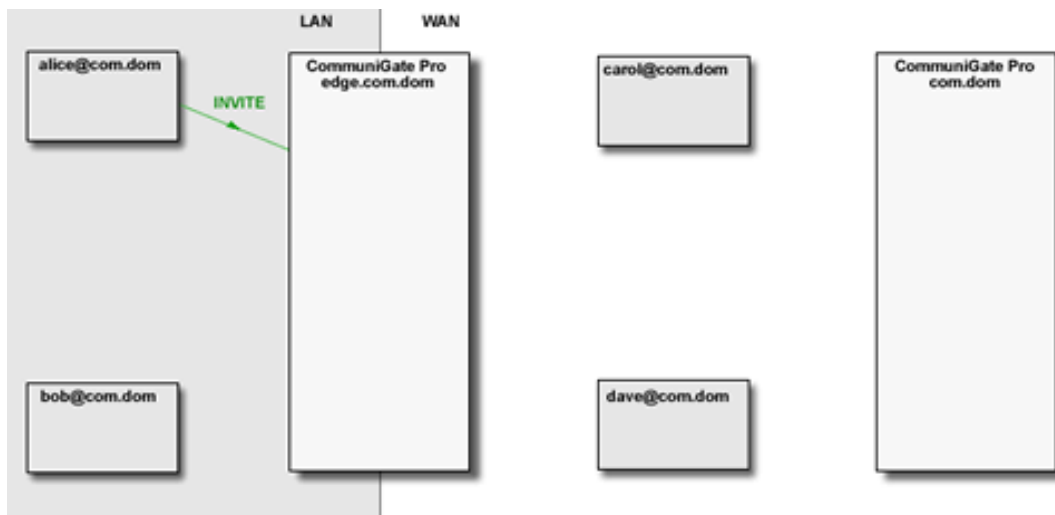
Step 5.



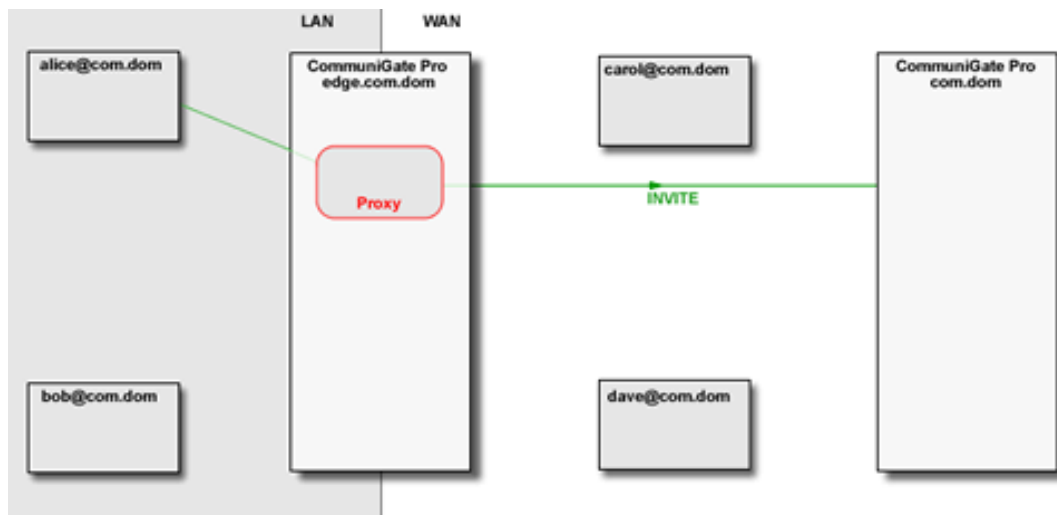
Step 6.



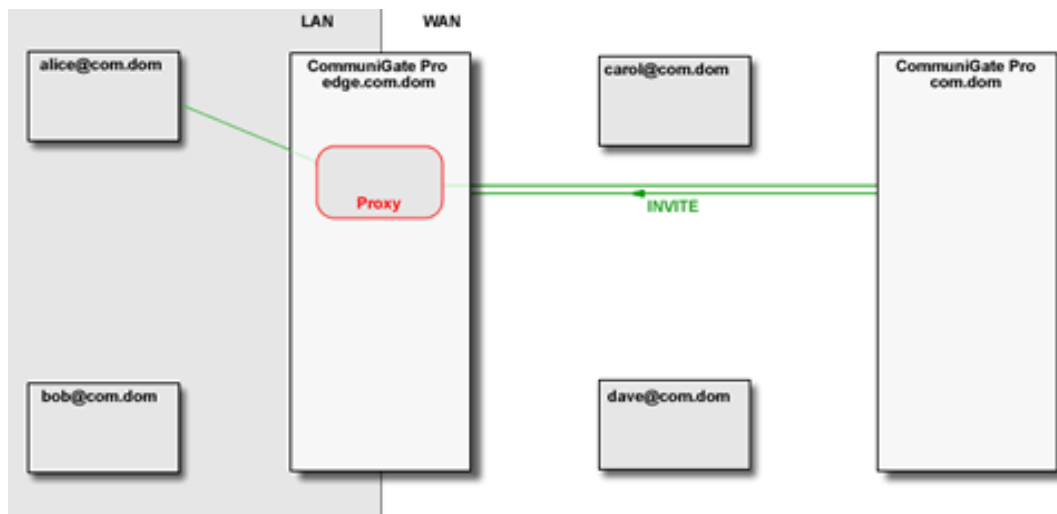
Step 7.



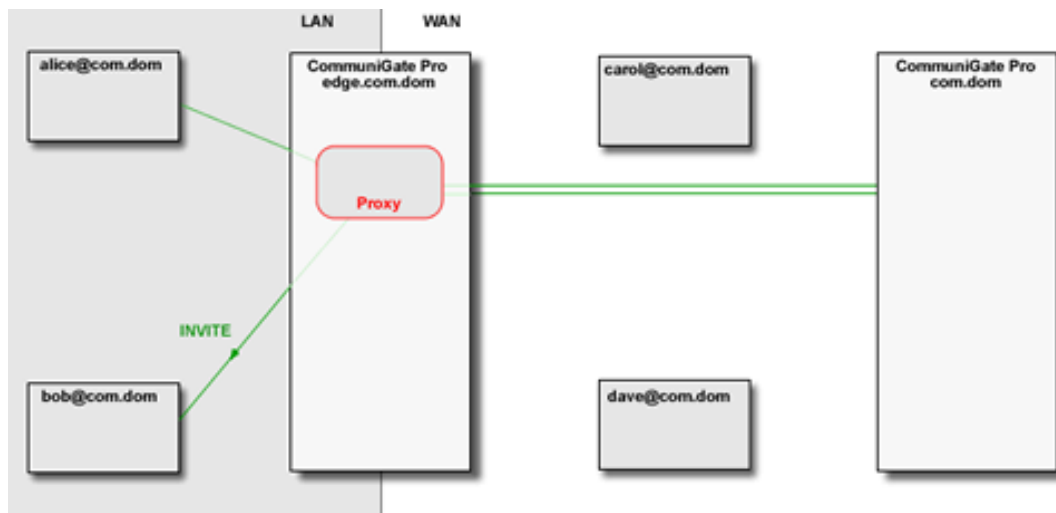
Step 8.



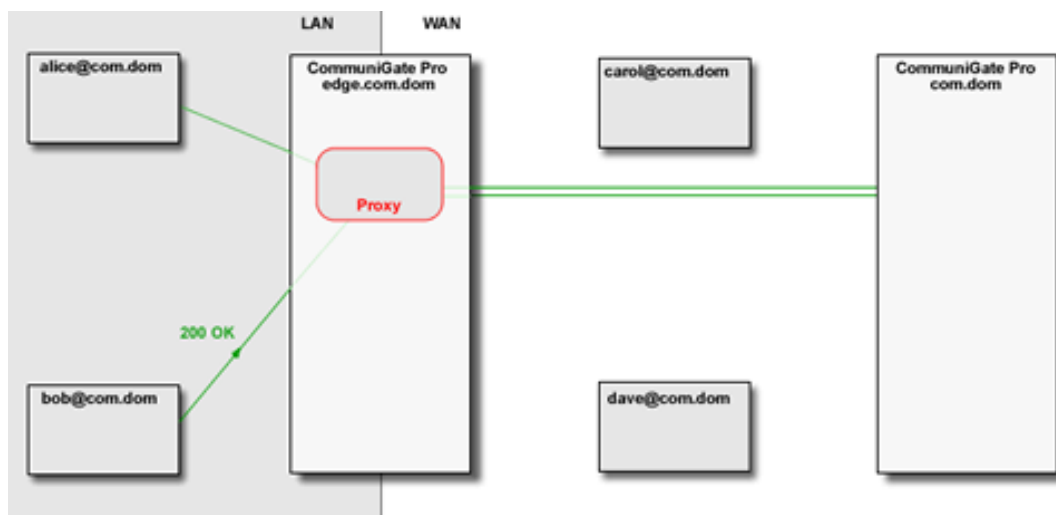
Step 9.



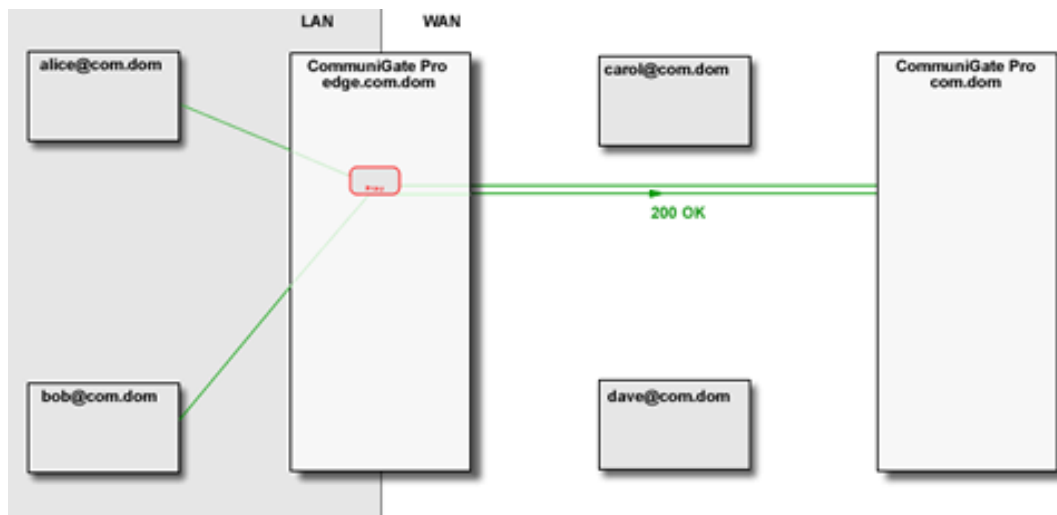
Step 10.



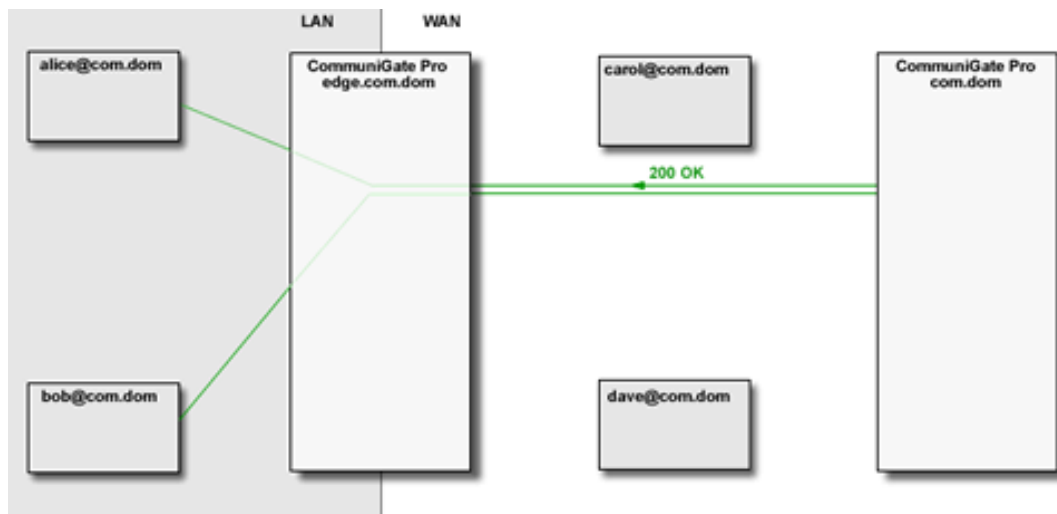
Step 11.



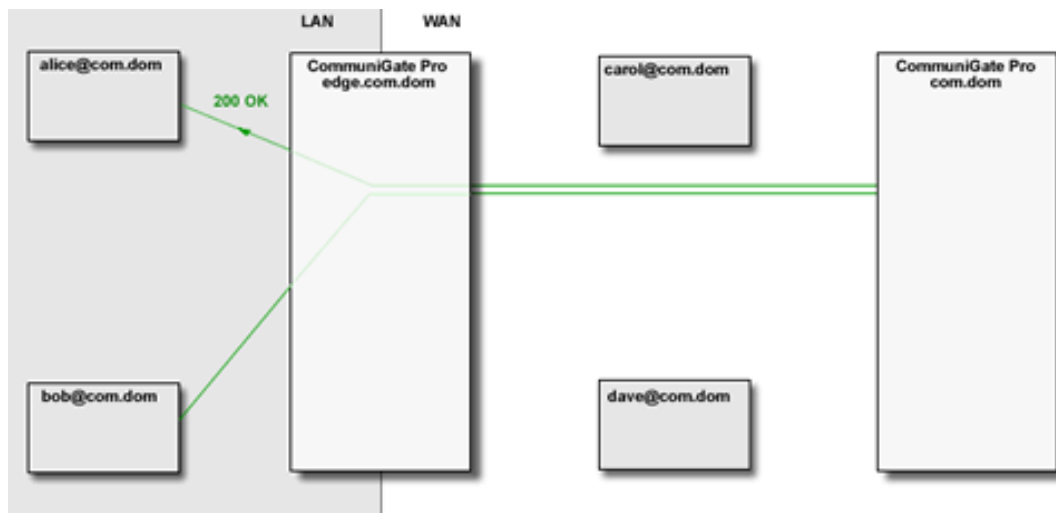
Step 12.



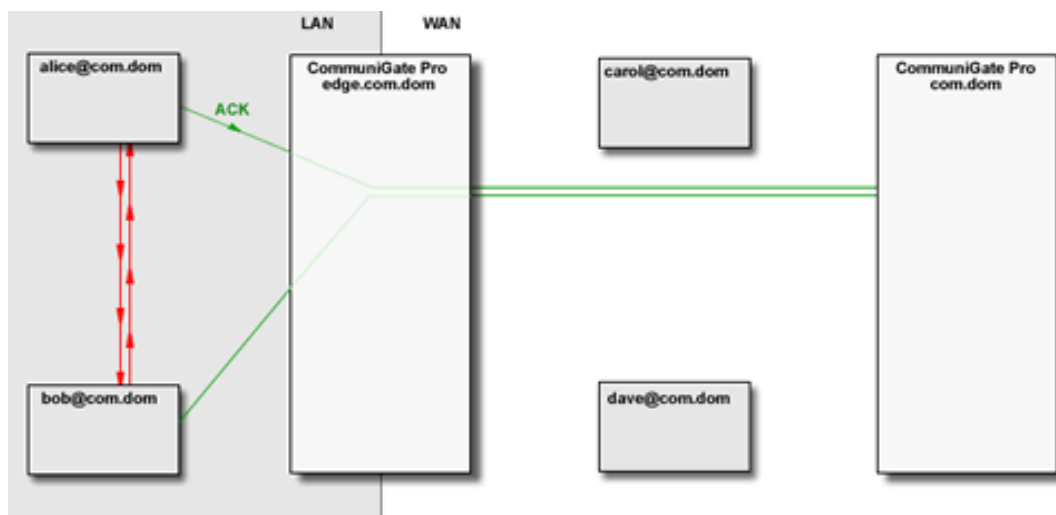
Step 13.



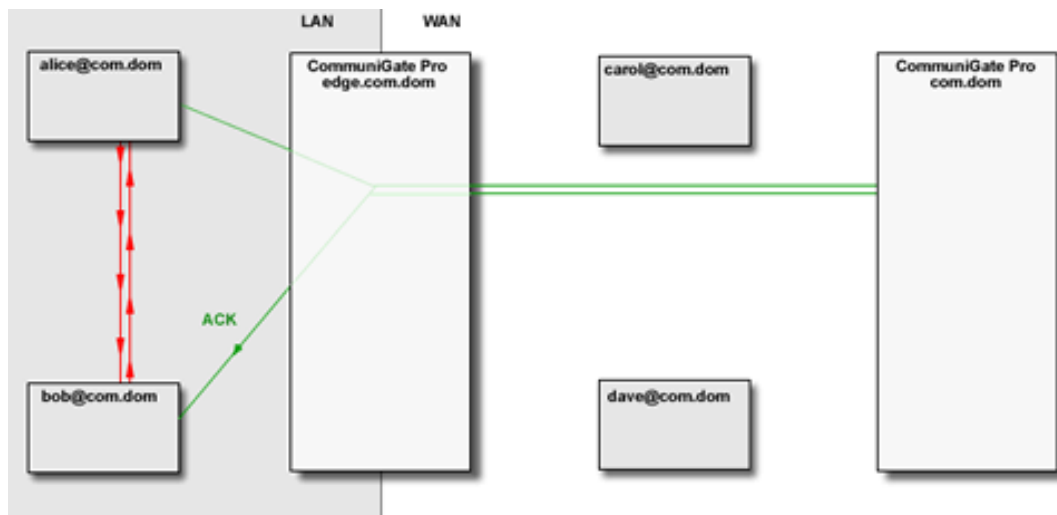
Step 14.



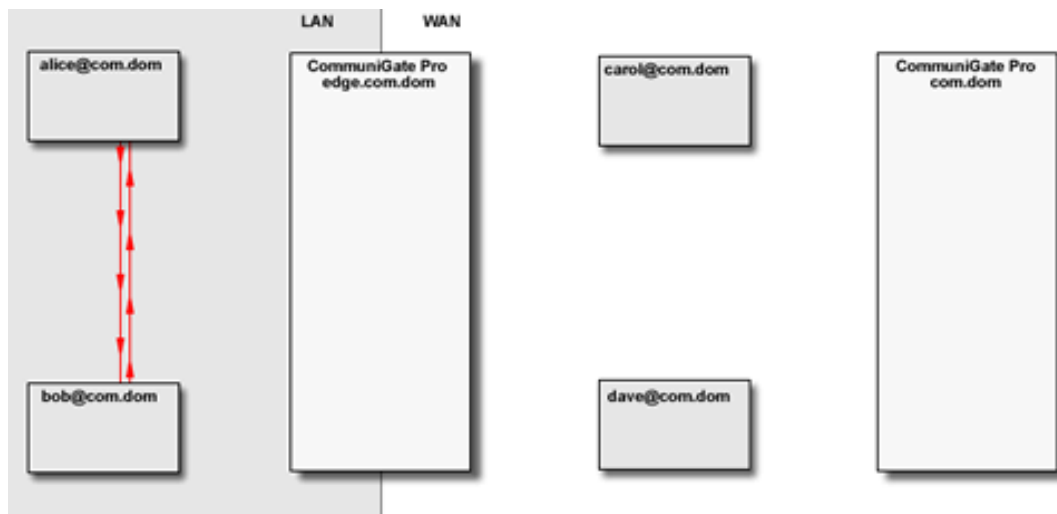
Step 15.



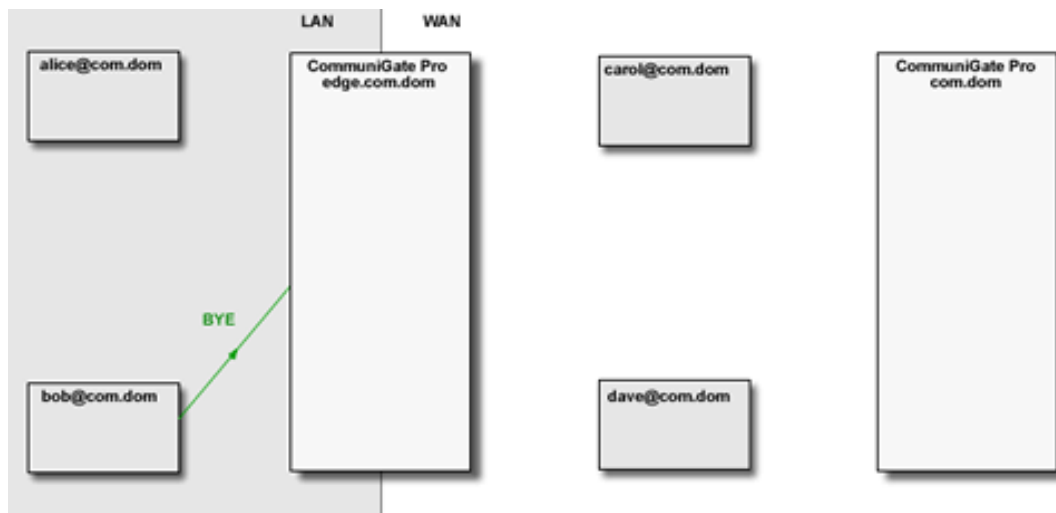
Step 16.



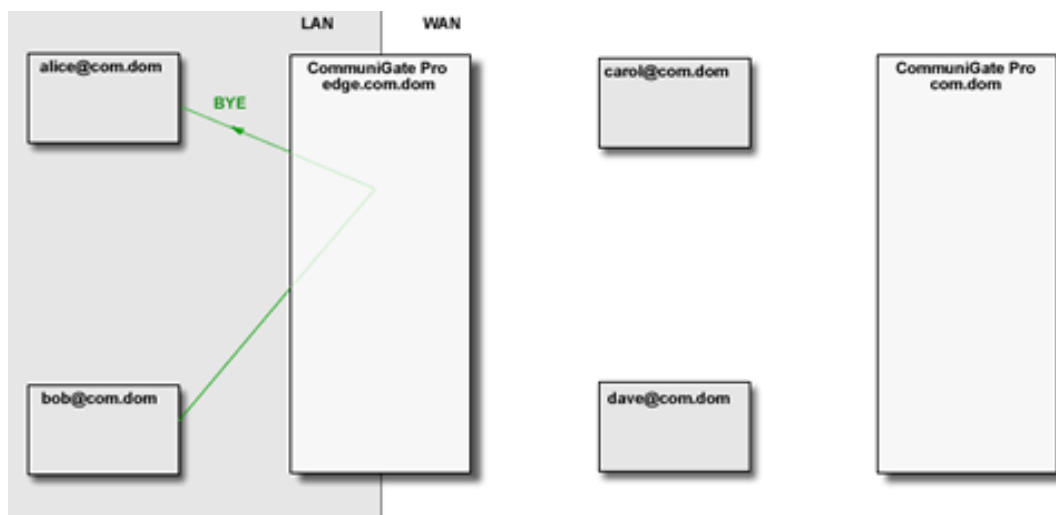
Step 17.



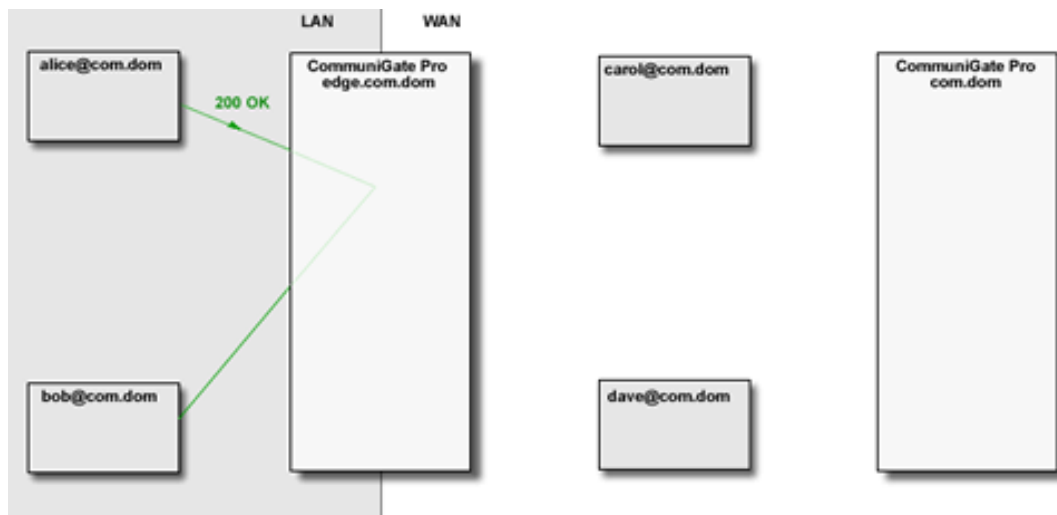
Step 18.



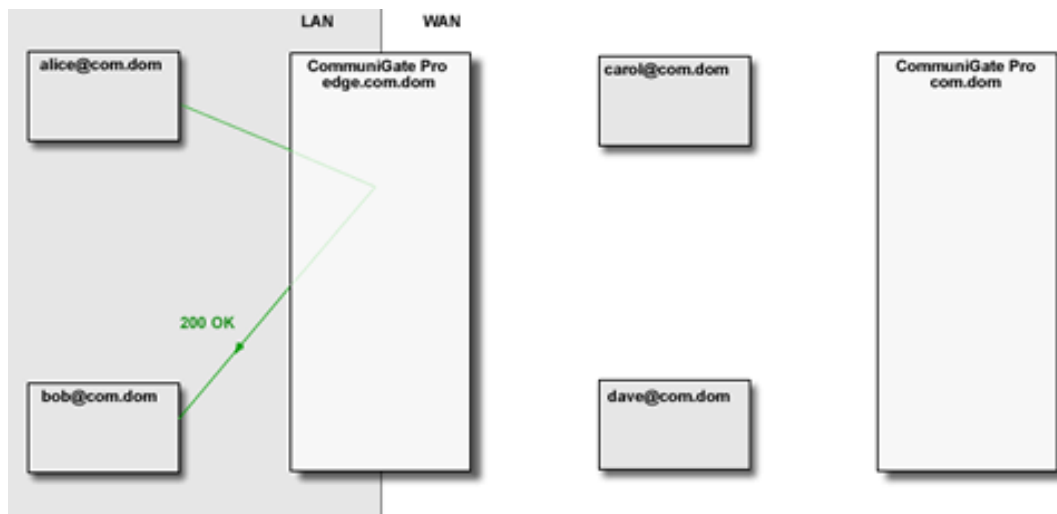
Step 19.



Step 20.



Step 21.



The SIP module can detect much more complex loop cases, either avoiding Media Proxies altogether, or minimizing the number of Media Proxies used.

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.

Open the SIP Module settings and click the Workarounds link. A table appears:

Problems/Bugs							
Agent Name	Microsoft	SubPresence	NoTCP	NoMaddr	NoPath	BadByeAuth	NeedsEpid
RTC/*	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SomeClient*	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

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							
Domain Name	Microsoft	SubPresence	NoTCP	NoMaddr	NoPath	BadByeAuth	NoSubMWI
sip.provider.dom	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

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 uses 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.

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.

The SIP module immediately accepts all signal addresses with domain names ending with the `.siggw` suffix. The rest of the domain name should specify the name of an [External Gateway](#) and the Signal is routed to that External Gateway.

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:

SIPS					Filter <input type="text"/>	Elements:2
ID	Status	Msgs	Phase	Source	Target	Info
38984	waiting (15s)	0	completed	udp[10.10.0.226:59645]	SIGNAL-40726	OPTIONS sip:ns.stalker.com
38994	waiting (7s)	0	completed	tcp[10.0.14.193:24777]	SIGNAL-40734	MESSAGE sips:user@stalker.com

The SIPC frame displays the active SIP Client transactions:

SIPC					Filter <input type="text"/>	Elements:2
ID	Status	Msgs	Phase	Source	Target	Info
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@stalker.com



XMPP Module

The CommuniGate Pro XMPP Module implements the [XMPP protocol](#) via IP networks.

The module is used to receive [Signal](#) Requests from remote entities, and to send Signals to remote entities.

Extensible Messaging and Presence Protocol (XMPP)

The CommuniGate Pro XMPP Module implements the XMPP protocol functionality.

XMPP Server Settings

To configure the XMPP module, use a Web browser to connect to the CommuniGate Pro Server WebAdmin Interface, and open the XMPP page in the Settings realm.

To configure the XMPP module, you should have the Can Modify Settings [access right](#).

Under construction

Routing

Under construction

Monitoring XMPP Activity

The Monitors realm of the [WebAdmin Interface](#) allows Server Administrators to monitor the XMPP module activity.

Under construction



PSTN

The CommuniGate Pro Server can use various *PSTN Gateways* to connect the modern VoIP network to the traditional Public Switched Telephone Network. 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.

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.

Call Routing

The CommuniGate Pro [Router](#) processes every Signal address in the system.

- The Router detects telephony (PSTN) type addresses, converts them to the standard E.164 form (+country_code area_code local_number) and routes them to a 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.

- The Router maps PSTN numbers to VoIP addresses, if possible. To map PSTN numbers, the Router can use global and ENUM services, as well as CommuniGate Pro Forwarders.
- The Router directs unmapped PSTN number either directly to a PSTN Gateway, or to a CommuniGate Pro Real-Time application designed to support external PSTN Gateways.

See the [Router](#) section to learn about the Default Router records.

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 domain name of the Gateway to use.

If this setting value is empty, the Account is not allowed to make PSTN calls and the call is 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 settings 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)

The "stock" gatewaycaller application does not use this setting.

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 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 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 by the Tasks themselves, some are relayed to the peering Task for relaying to the caller or the gateway.

The technology used to implement this type of call setup (when a call is established using two formally independent Signalling 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 asterisk (*) 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.

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 `localAreaCode` application.

The stock `localAreaCode` application requires the caller to authenticate, and then it retrieves the following Account [PSTN Settings](#):

`PSTNAreaCode (Local Area Code)`

This settings 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)@*> = localAreaCaller{*}#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, concatenates 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:

```
<0(7-20d)@*> = +49*@telnum
```

Note: this record should be specified AFTER the

```
<00(8-20d)@*> = +*@telnum
```

record that handles internation phone calls.

You can also use the `localAreaCaller` application to properly route Domestic calls. Route all calls with the domestic prefix to the `localAreaCaller` application, but specify the second parameter - a string `c`:

```
<0(8-20d)@*> = localAreaCaller{*,c}#pbx@localhost
```

The `localAreaCaller` 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 settings specifies 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).

Incoming Calls

PSTN Gateways can direct incoming calls to the CommuniGate Pro Server. There incoming Gateways usually have almost the same problems as the outgoing Gateways: inability to transfer calls, inability to switch media streams, etc.

If incoming calls are routed to the CommuniGate Pro [PBX](#) application, that application acts as a B2BUA and addresses these issues.

If incoming calls are directed to individual CommuniGate Pro [Accounts](#) (or other [Objects](#)), a direct session between the Gateway and the end-user device can be established, and all Gateway problems will be exposed to the end-user.

To solve PSTN Gateway problems when directing calls to addresses that are not necessarily answered with CommuniGate Pro Real-Time Applications, you may want to route all these calls to a special `gatewayincoming` application. This application, acting as B2BUA, will create a separate Task ("call leg") to call the destination and it will bridge the call in the same way the `gatewaycaller` application processes outgoing calls.

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 this 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
```

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)
	 <input type="text" value="1(408)"/>
Gateway Name	 pstn.provider.com
	 <input type="text" value="personal.provider.com"/>
Gateway Address	
	



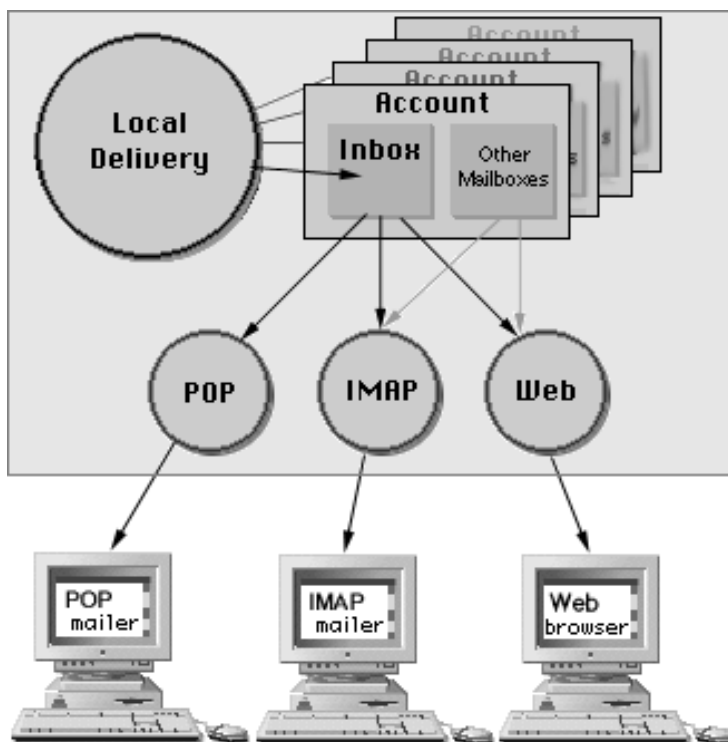
Account Access

The CommuniGate Pro Server allows users to use various mailer applications to access their accounts and mailboxes.

- The [POP module](#) is a POP3 server that allows users to retrieve mail from their INBOX mailboxes using the POP-based mailers.
- The [IMAP module](#) is an IMAP4rev1 server that allows users to process mail messages in all account mailboxes using IMAP-based mailers.
- The [WebMail module](#) is an HTTP (Web) server that allows users to process mail messages in all account mailboxes using any Web browser.
- The [XIMSS module](#) is an [XML Messaging, Scheduling, and Signalling](#) 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](#) allows users to access their accounts and mailboxes the Microsoft® Windows MAPI (Mail API) and use Microsoft Outlook in the "groupware" mode.
- The [FTP module](#) is an FTP server that provides access to [Personal File Sites](#).
- The [TFTP module](#) is a TFTP server; it provides access to Account [File Storage](#).
- The [ACAP module](#) is an ACAP server that allows users to manage their Accounts.

Access to Accounts

Every CommuniGate Pro Account can be accessed via Access modules - POP, IMAP, Web Email, 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](#): an Account mailbox can be accessed not only by the Account owner, but by other Account 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 several domains. In order to do this, the server should get the name of the domain name in which to look for the specified account. As for mail delivery, the server needs the "full account name" i.e. an address in the form *accountname@domain-name*.

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 via a POP or IMAP mailer, it is possible to specify the full account name in the mailer "account name" settings. Since many mailers do not like to see the @ symbol in the account name, the % symbol can be used instead. The user john that has an account in the secondary domain client1.com should specify the account name as `john%client1.com`, not just as john.
- 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 a dedicated IP address assigned to that domain, and a user tries to connect to the server via that IP address, all simple account names specified with the user mailer are processed as account names in that domain.
Additional IP addresses can be rather expensive, so this method should be used only if it is impossible to make users specify domain names explicitly.

All methods can be mixed in one server: 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 mailer 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 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 [dedicated to some secondary domains](#). If a connection IP address is dedicated to a secondary 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 an account in 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 POP and IMAP mailers to access an account `info`, but they specify different names in their "mail 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 mailer takes the specified "mail 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 mailer 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 the main domain name points to that IP address, 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 checking mail:

- The mailer takes the specified "mail 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 mailer 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 the `client2.com` secondary domain name points to that IP address, so it adds the secondary domain name `client1.com` to the specified simple name.
- The server gets the correct full account name `info@client1.com`.

When the second user starts checking mail:

- The mailer takes the specified "mail 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 mailer 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 the main domain name points to that IP address, so it adds the main domain name `company.com` to the specified simple name. The `client2.com` domain name also points to that IP address, but the server dedicates IP addresses to only one domain, and it always assigns it to the first domain it processes, the main domain `company.com` in our case.

- The server gets the **incorrect** full account name `info@company.com`.

This happens because the mailer has not passed the information about the "mail server" name from its settings, and the only information the Server has is the IP address. But since the IP address is the same for both Main Domain `company.com` and the secondary Domain `client2.com`, the Server is unable to detect which Domain is needed and defaults to the main domain.

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 users starts the mailer:

- The mailer takes the specified "mail 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 mailer 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@company.com`.

This problem does not appear if the third user uses the Web Interface: the server always gets the addressed domain name via the HTTP protocol, so it does not have to detect that name by looking at the IP addresses.

Note: 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 `%` signs, the Connector adds the `'@'` sign and `Server Name` setting value to the specified account 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 addresses is also applied to the account names specified with mailer applications.

Sample:

The account `John` has an alias `John_Smith`;
all E-mail messages addressed to `John_Smith` will be stored in the account `John`;
the user can specify either `John` or `John_Smith` as the "account name" setting in his mailer - in both cases the account `John` will be opened when his mailer starts a session.

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 an alias record with the Router:
`<John> = j.smith@client1.com`;
all E-mail messages addressed to `John@company.com` will be stored in the account `j.smith` in the secondary domain `client1.com`;
the user can still specify just `John` as the "account name" setting, and the same `company.com` "mail server" setting in his mailer - but the server will open the account `j.smith` in the `client1.com` domain.

Note:

do not create an alias record that redirects the Postmaster account in the main domain. You will not be able to administer the server, if the postmaster account is redirected to a nonexistent Account or to an Account that does not have the postmaster access rights.

If you want the postmaster mail to be directed to some other user, do not use the Router, but use the postmaster account [Rules](#) instead.



Mailbox Sharing

The word "shared" is used to describe several different features a messaging server can provide. Because of that, this CommuniGate Pro Guide uses the following terms:

- *Simultaneous Access* - a situation when several client applications work with the same mailbox at the same time.
- *Foreign Mailbox Access* - a situation when an Account user works with a mailbox created in a different Account.
- *External Mailbox* - a CommuniGate Pro mailbox located outside the CommuniGate Pro *base directory* and modified directly by other programs that do not use messaging protocols and bypass the CommuniGate Pro Server.

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 (POP, IMAP, Web, etc.) 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](#), or [WebUser](#) 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 it lets a group of people (i.e. the sales department) process messages in one centralized mailbox.

Foreign and Public Mailboxes

The CommuniGate Pro access system 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 Control Lists](#).

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 Web User 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 submailboxes, 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) do not support foreign mailboxes at all. To let those clients access shared mailboxes in other Accounts, [Mailbox Aliases](#) can be used.

External 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 account mailboxes. Instead, those programs read and modify mailbox files directly, via the file system.

The CommuniGate Pro allows you to create accounts with *external* INBOX mailboxes. These mailboxes are stored not inside the CommuniGate *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 the 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 external mailboxes only when absolutely necessary, because:

- access to external mailboxes is less effective because of the 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 have been deleted from a mailbox during the time a local mailer was active.

The bugs in the local mailers *have nothing to do with the CommuniGate Pro Server*, and they can result in mailbox corruption on any system. You may check for yourself - sample shell sessions show you how the Unix `mail` programs can destroy your mailbox.

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 [External Mailboxes settings](#) for that domain.

When you create an account that has an external 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 external INBOX, the Server does NOT remove the INBOX mailbox file.



POP Module

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 messages to the CommuniGate Pro server.

The POP module activity statistics is available via the CommuniGate Pro [SNMP](#) agent.

Post Office Protocol (POP3)

The Post Office Protocol allows computers to retrieve messages from mailboxes on 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, 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.

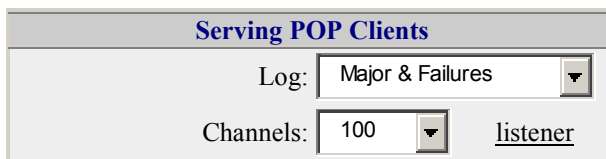
The POP3 protocol does not support multi-mailbox accounts. If a client application specifies a multi-mailbox (folder) account, the INBOX mailbox is opened.

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 a Web browser to configure the POP module. Open the Access page in the WebAdmin Settings section (realm):



Serving POP Clients	
Log:	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.

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 [Direct Mailbox Addressing](#) 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 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.

In order to support Microsoft® E-mail clients, the POP module supports the non-standard "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 mailbox *mailboxname* in the *accountname* account, a user should specify the account name as: *mailboxname#accountname*:

Account name (specified in the mailer settings)	Accessed Mailbox
<code>jsmith</code>	mailbox INBOX in the <code>jsmith</code> account
<code>private#jsmith</code>	mailbox <code>private</code> in the <code>jsmith</code> account
<code>lists/info#jsmith@client1.com</code>	mailbox <code>lists/info</code> in the <code>jsmith</code> account in the <code>client1.com</code> 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 [Direct Mailbox Addressing](#) 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
<code>jsmith</code>	mailbox INBOX in the <code>jsmith</code> account
<code>~public/announces#jsmith</code>	the public mailbox <code>announces</code>
<code>~boss/INBOX#jsmith</code>	mailbox INBOX in the <code>boss</code> 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 C11, 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 C11 account mailbox.

If a user connects as C11, 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 = C11.local
```

The first message is sent to:

```
abcdef@client1.com
```

It is stored in the C11 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 C11 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 C11
```

```
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

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.

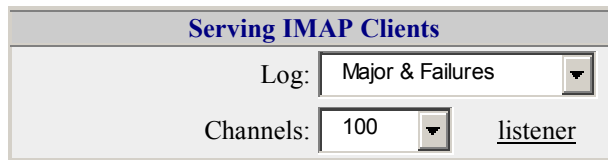
Internet Message Access Protocol (IMAP)

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 a Web browser to configure the IMAP module. Open the Access page in the WebAdmin Settings section (realm):



Serving IMAP Clients	
Log:	Major & Failures
Channels:	100
listener	

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.

MultiAccess

While many other IMAP servers "lock" opened mailboxes, the CommuniGate Pro IMAP is designed to provide simultaneous access to any mailbox for any number of clients.

In reality, the IMAP module uses the CommuniGate Pro Mailbox Manager that provides this feature for all types of clients. See the [Sharing](#) 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 [Sharing](#) 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 (in the account settings). 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 Web User 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 (such as Microsoft Outlook and Outlook Express) 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 [Advertise Clear Text methods](#) 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.

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:

IMAP Server <input checked="" type="checkbox"/>					Channels:3
ID	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.

The IMAP activity statistics is available via the CommuniGate Pro [SNMP](#) agent.

IMAP Implementation Details

The CommuniGate Pro IMAP module implements many IMAP extensions. Some of these extensions work in the implementation-specific manner.

QUOTA

Each account has its own Quota Root - `#Account`. It is possible to read other account quotas if the currently authenticated IMAP user has the `Domain Administrator` access right or the `Can Modify Accounts and Domains Server Administrator` access right.

It is possible to set other account mail store quota using the `SETQUOTA` command if the currently authenticated IMAP user has the `MaxAccountSize` `Domain Administrator` access right or the `Can Modify Accounts and Domains Server Administrator` access right.

NAMESPACE

The standard CommuniGate Pro "account name" prefix - the tilda sign (~) can be used to access mailboxes in other Accounts.

The `~public` prefix is reported as the `Public Namespace` prefix, but it's the job of the Server/Domain Administrator to ensure that the Account `public` is created in the proper Domain.

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 Extension standards.

UNSELECT

This IMAP command is equivalent to the CLOSE command, but it does not expunge any message marked as \Deleted

Additional STATUS data items.

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 summ 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
```

Additional LIST extensions.

The LIST command can use additional options along with the options specified in the LISTTEXT extension standard:

UIDVALIDITY, 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 UNSEEN INTERNALSIZE) "" "ma%"
* LIST (\HasNoChildren \UNSEEN(14) \INTERNALSIZE(2345678) \Unmarked)
mailbox
A001 OK completed
```

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".



WebUser Interface

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. So, 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 will display the `client.com` Login page, too.

If the URL specifies a domain that is not among the main and secondary server 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 will not get this request), but that name is not routed to any of the secondary domains on your server. You should either create a secondary domain with that name, or route this domain to one of the existing CommuniGate Pro domains.

If a URL specifies an IP address instead of a domain name, the WebUser module tries to find a secondary domain to which the specified address is dedicated. If no secondary 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 Server 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

IMAP and POP are session-oriented protocols: a client mailer 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 session-oriented: 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 works 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. Usually, users of large providers (such as AOL, WebTV) access the Internet via the provider's proxy servers, so their accounts should have the address-controlling option disabled.

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.

WebUser Interface Settings

To configure the WebUser Interface module, use any Web browser to connect to the CommuniGate Pro Server, and open the WebUser page in the Settings realm. To configure the WebUser Interface module, you should have the Can Modify Settings access right:

Sessions		Limit:	3000
Log:	Major & Failures	Inactivity Time-Out:	30 minutes
Compose Limit: Recipients:	50	Session Time Limit:	6 hours

Limit

Use this setting to specify the maximum number of concurrent WebUser Interface Settings.

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 Web Session. This settings 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.

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 :

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.

Enable	Language	Log Level	Program Name and Parameters
<input checked="" type="checkbox"/>	English-GB	Major & Failures	<div>/usr/local/bin/aspell -a --lang=en_GB</div> <div>Convert to: ISO-8859-1</div>
<input checked="" type="checkbox"/>	English-US	Major & Failures	<div>/usr/local/bin/aspell -a --lang=en_US</div> <div>Convert to: ISO-8859-1</div>
<input checked="" type="checkbox"/>	Russian	Major & Failures	<div>/usr/local/bin/aspell -a --lang=ru</div> <div>Convert to: KOI8-R</div>
<input type="checkbox"/>		Major & Failures	<div></div> <div>Convert to: ISO-8859-1</div>

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 a 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. Since mailing lists are archived in mailboxes, the mailing list WebUser interface is similar to the Mailbox Browsing interface.

The mailing list Web User 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.



XIMSS Module

The CommuniGate Pro XIMSS module implements an open [XML API](#) 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.

See the [XML API](#) section for more details on the protocol and its features.

Configuring the XIMSS Module

Use a Web browser to open the Access page in the WebAdmin Settings section (realm):

XIMSS Server	
Log:	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 XIMSS 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 mail 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 modules 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 settings all its ports to zero.

Flash Security

When a Flash client connects to an XIMSSocket server (such as the 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.

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:

XIMSS Server <input checked="" type="checkbox"/>					Channels:3
ID	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.

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.

The XIMSS activity statistics is available via the CommuniGate Pro [SNMP](#) agent.



MAPI Connector

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 MAPI Connector.dll) should be installed on client Microsoft Windows workstations.

The CommuniGate Pro MAPI Connector acts as a "MAPI provider". It accepts Messaging API requests from Microsoft Outlook (Outlook 98, Outlook 2000, 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.

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.

The CommuniGate Pro MAPI Connector requires either a Groupware-type License Key or a special MAPI License Key. The Server accepts up to 5 concurrent MAPI sessions without these Keys.

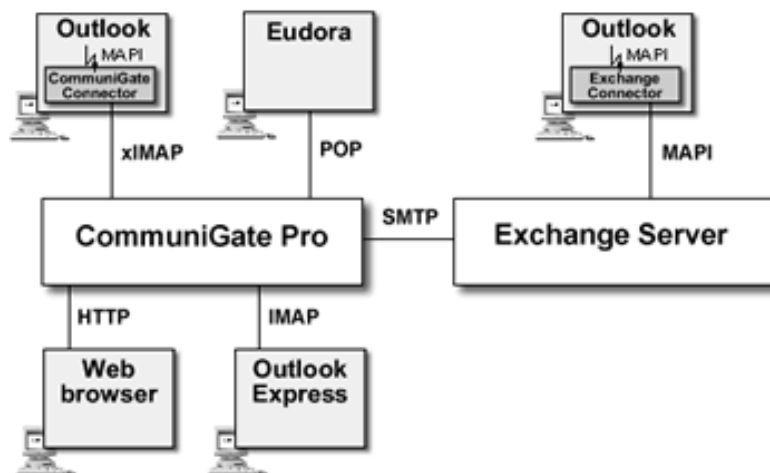
MAPI Connector Overview

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).

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 or the CommuniGate Pro WebUser Interface:



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 `cgmxi32.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.

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".

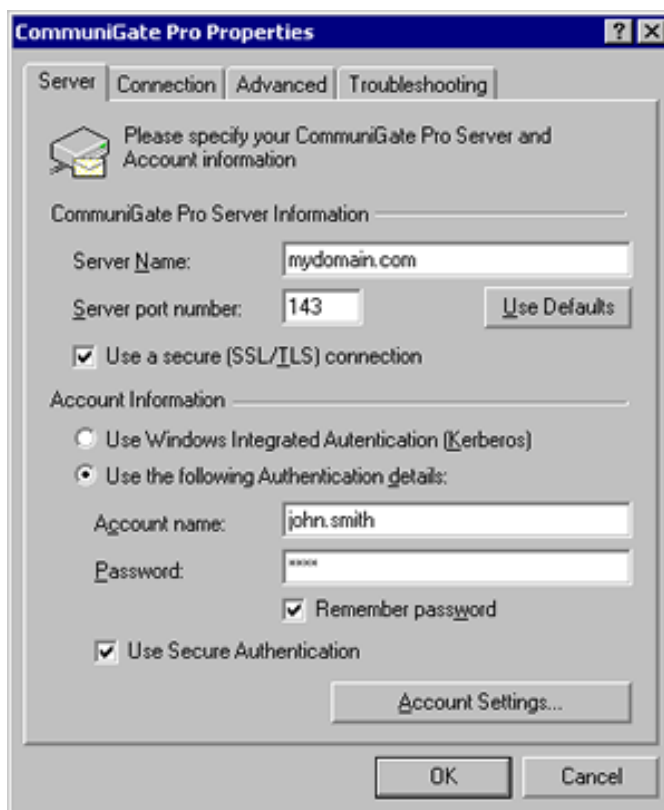
You can add other services into the same Profile.

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:



The image shows the 'CommuniGate Pro Properties' dialog box with the 'Server' tab selected. The dialog has four tabs: 'Server', 'Connection', 'Advanced', and 'Troubleshooting'. The 'Server' tab contains the following fields and options:

- CommuniGate Pro Server Information:**
 - Server Name:** mydomain.com
 - Server port number:** 143 (with a 'Use Defaults' button)
 - ☒ Use a secure (SSL/ILS) connection
- Account Information:**
 - ☐ Use Windows Integrated Authentication (Kerberos)
 - ☒ Use the following Authentication details:
 - Account name:** john.smith
 - Password:** [masked]
 - ☒ Remember password
 - ☒ Use Secure Authentication
 - Account Settings...** button

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

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 Connect 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 defaults

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.

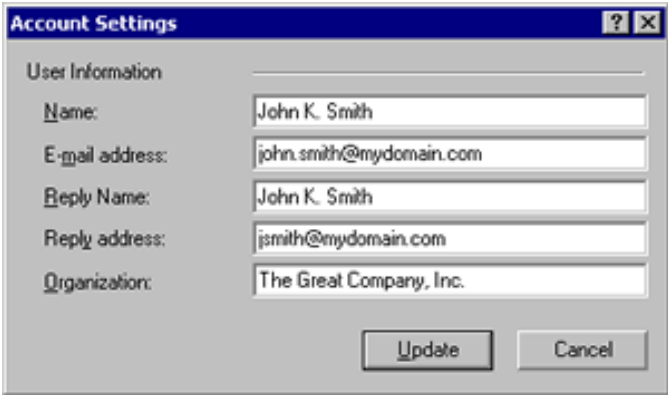
Use Secure Authentication

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

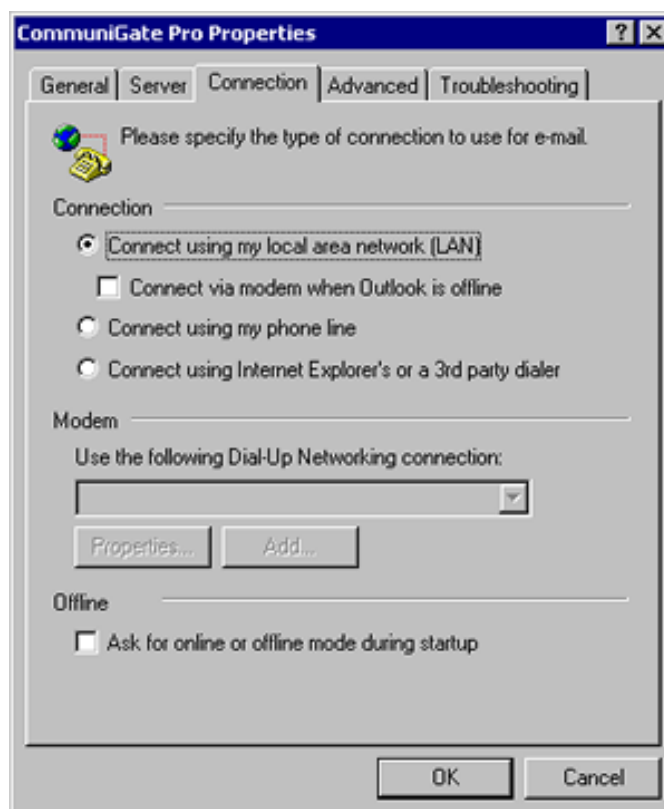
The Account Settings dialog box can be opened by pressing the Account Settings button on the Server panel. The box allows you to specify the MAPI Account name and other general data:



The image shows a Windows-style dialog box titled "Account Settings". It has a blue title bar with a question mark icon and a close button (X). The dialog is divided into two main sections. The top section is labeled "User Information" and contains five text input fields. The first field is labeled "Name:" and contains the text "John K. Smith". The second field is labeled "E-mail address:" and contains "john.smith@mydomain.com". The third field is labeled "Reply Name:" and contains "John K. Smith". The fourth field is labeled "Reply address:" and contains "jsmith@mydomain.com". The fifth field is labeled "Organization:" and contains "The Great Company, Inc.". At the bottom right of the dialog, there are two buttons: "Update" and "Cancel".

Connection

The Connection panel allows you to select your network connection method.



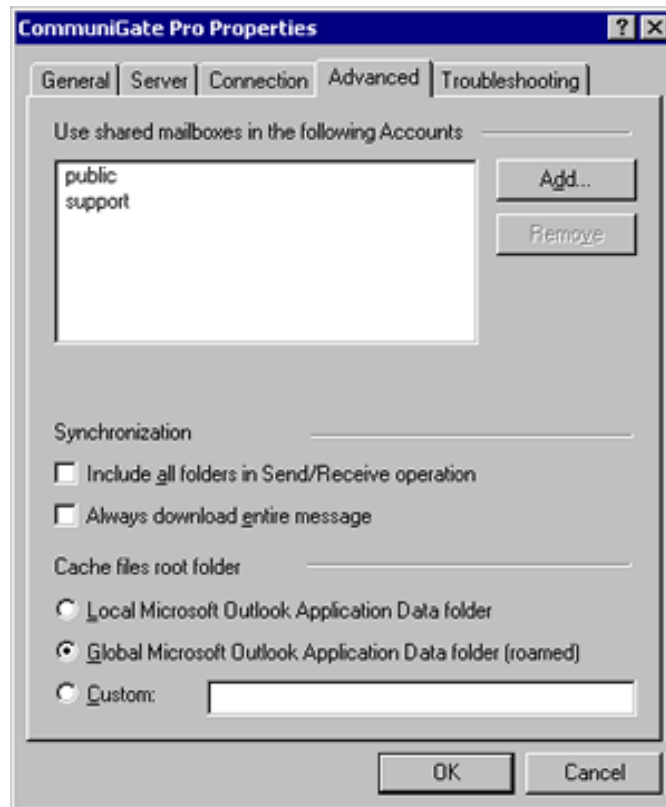
Offline

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 [Sharing](#) 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 if 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.

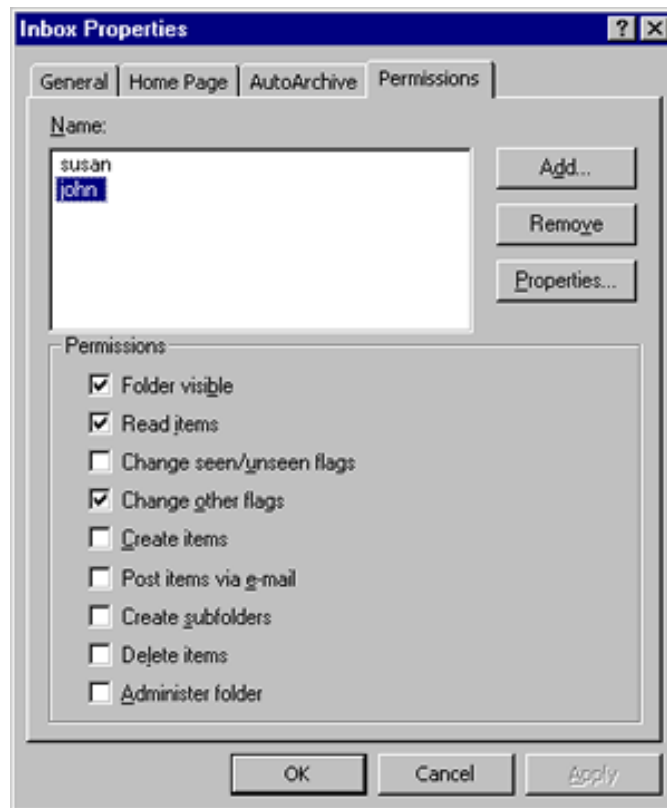
Custom

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.

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 the [Personal File Site](#) area. Publicly available information in the standard vCalendar format is stored as the `freebusy.vfb` file in the topmost directory of your Personal File Site.

Note: Make sure your CommuniGate Pro Account limits allow the MAPI Connector to store the file with your Free/Busy information in your Personal File Site area.

This feature allows users of Outlook and other calendaring clients to access your Personal File Site 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 not implemented via the MAPI Connector and should be done by the Outlook application itself. To configure your Outlook application:

- Install the Microsoft Web Publishing Wizard (it can be downloaded from the www.microsoft.com Web site).
- 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.

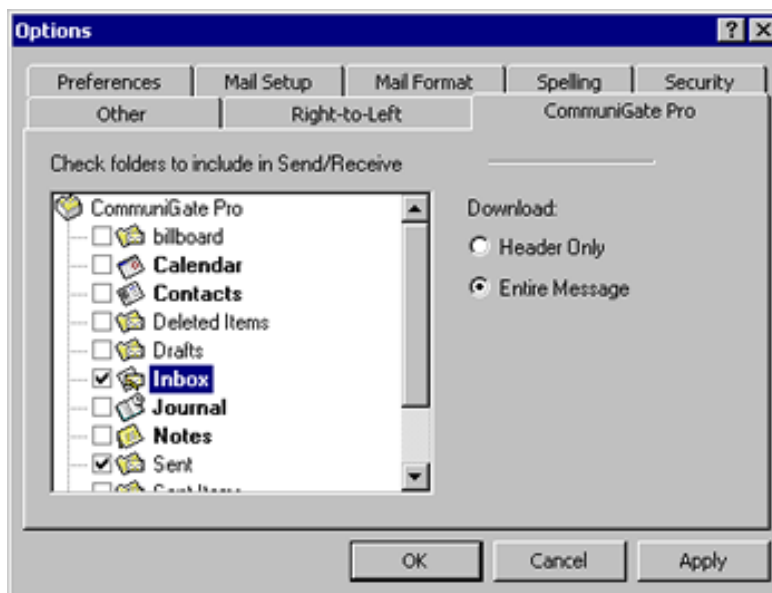
Search URLs specified above allow users to retrieve Free/Busy information for the users of the same CommuniGate Pro system.

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

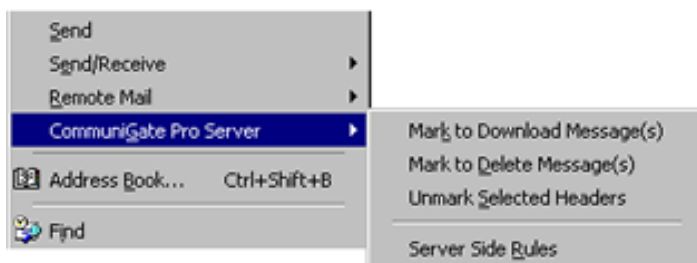
When you work in the Offline mode, the MAPI Connector does not have access to the messages stored on the CommuniGate Pro server. To be able to work productively, you need to make sure that the messages you need are stored in the MAPI Connector local cache. You specify the cache options on the per-mailbox (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 synchronise the changes you do in the Offline mode with the CommuniGate Pro server.



Synchronisation takes place when the Send/Receive operation is initiated (manually or automatically, based on a 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.

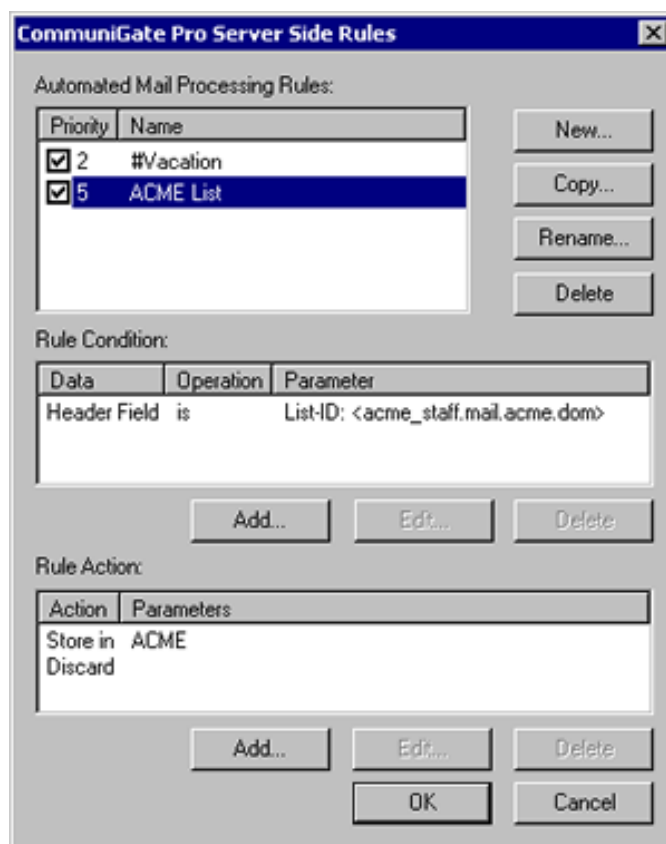
Unmark Selected Headers

Cancels the pending off-line operations for the messages selected in the Outlook mailbox view.

Configuring Automatic Rules

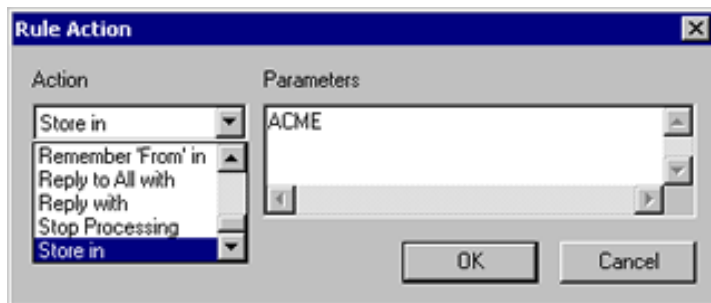
The MAPI Connector allows you to specify the server-side Rules to process mail coming to your Account.

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 applica-

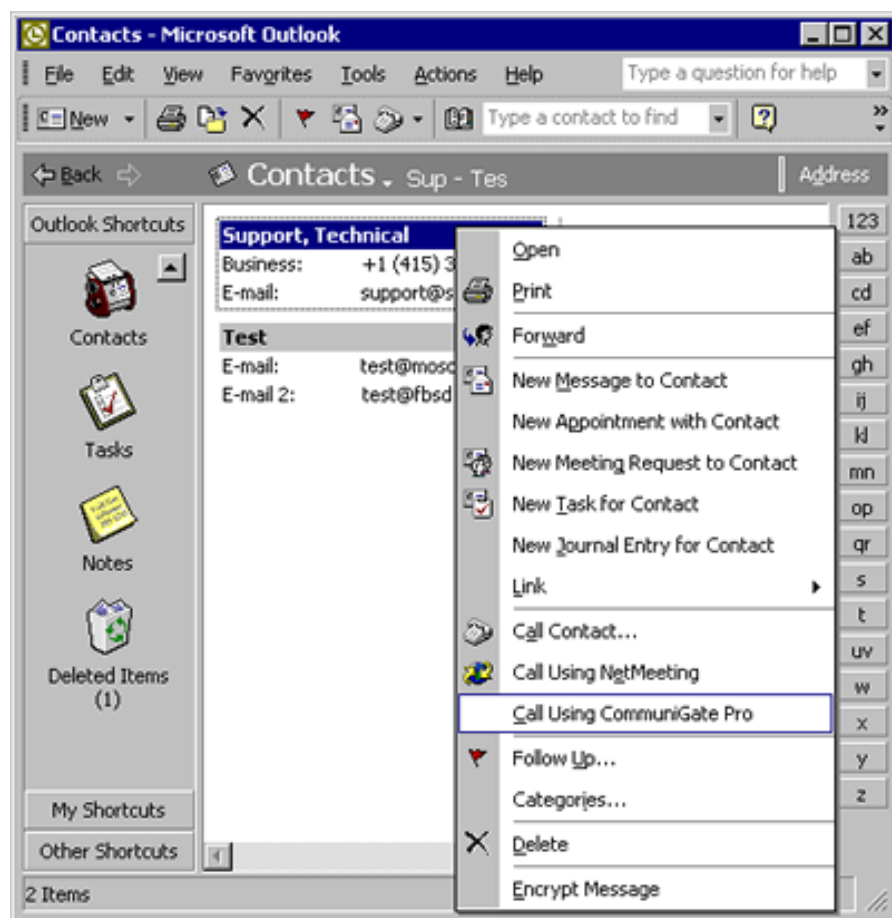
tions:

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

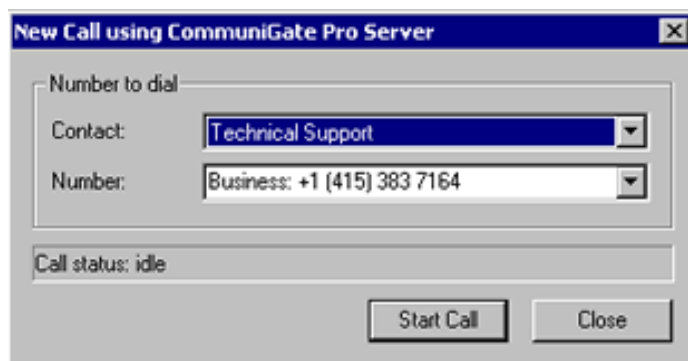
The MAPI Connector allows Outlook users to employ Real-Time (VoIP, etc.) functions of the CommuniGate Pro Server.

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.

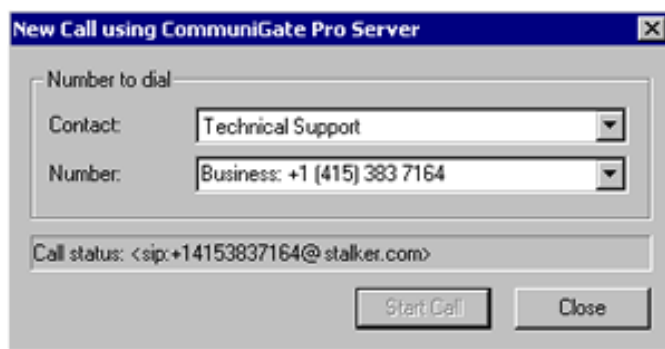
A dialog box with the Contact name and phone number appears:



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.

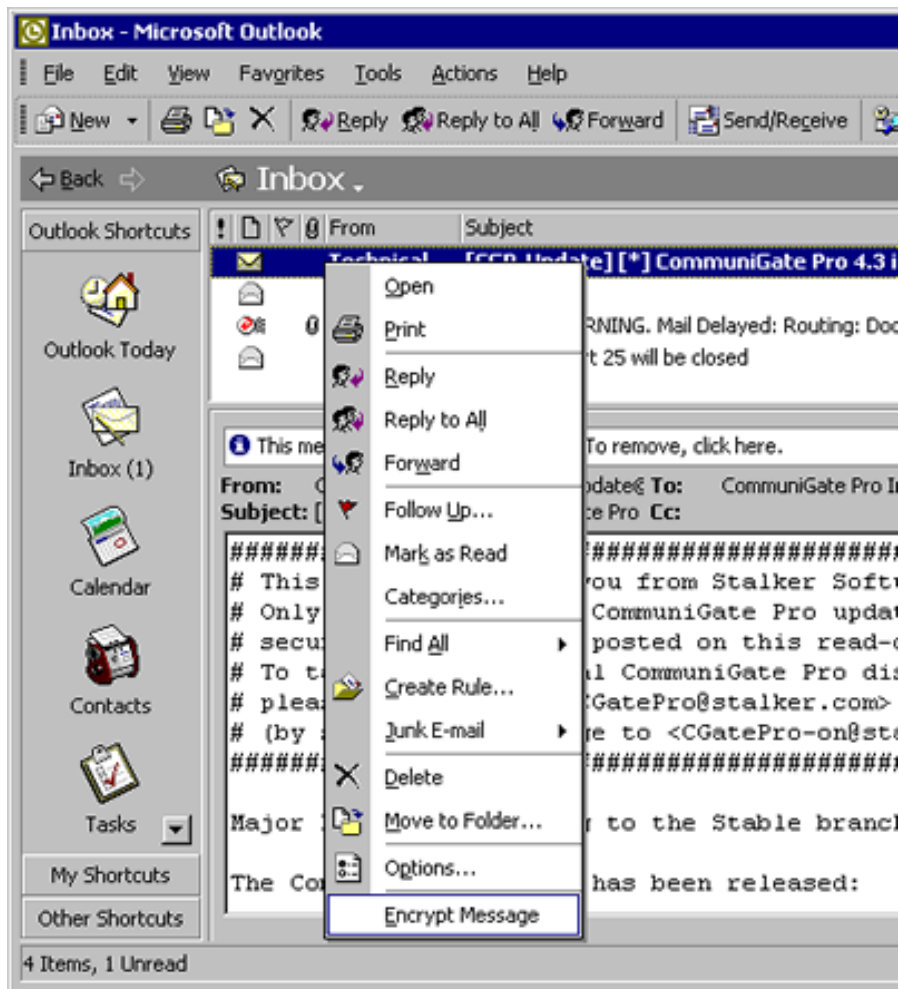
The dialog box has a field displaying the call status:



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

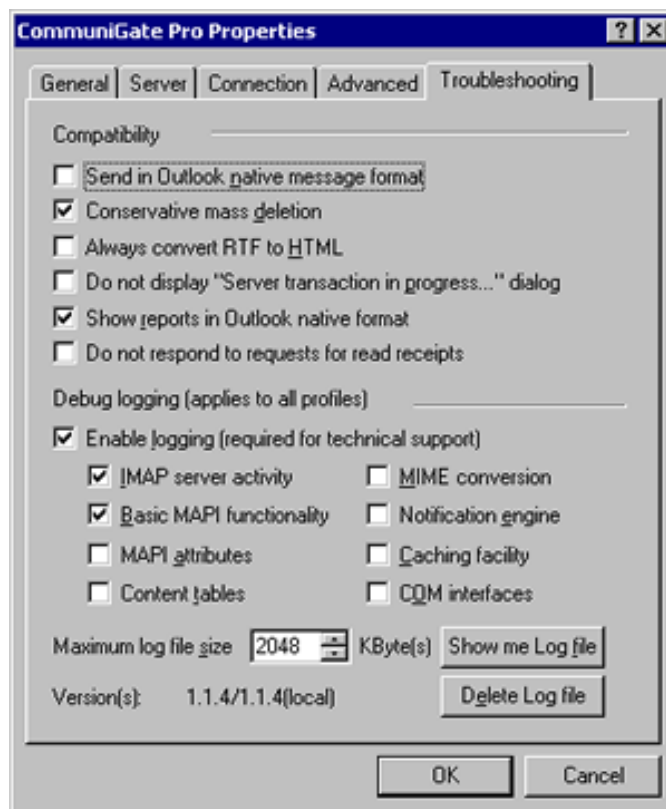
stored on the Server in the encrypted S/MIME format - as if the message sender has sent it encrypted.

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 [Stalker 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 [Stalker 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 may be interpreted incorrectly by Outlook Exchange (or Outlook IMAP) users. Checking this option may help to solve the problem.

Conservative mass deletion

In some configuration the Outlook Autoarchive function may remove messages in large portions. To avoid that 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.

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 client (Outlook) functions may be unavailable. Stalker Software 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.



FTP Module

The CommuniGate 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).

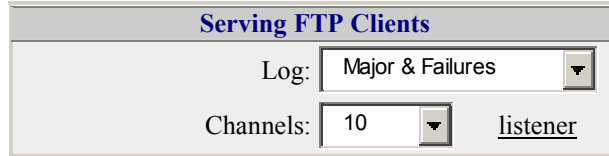
The CommuniGate Pro FTP module supports the REST command and it can resume broken file transfer operations.

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 a Web browser to configure the FTP module. Open the Access page in the WebAdmin Settings section.



Serving FTP Clients	
Log:	Major & Failures
Channels:	10
listener	

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. Since the FTP clients send passwords in the clear text format, setting the Log to these setting 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).

Access to Account File Storage

As soon as 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. >BR>The FTP module also 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.



TFTP Module

The CommuniGate 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. The CommuniGate Pro TFTP clients can retrieve data stored in Account [File Storage](#).

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 retrieve and the transfer mode. In return, the Server computer sends a UDP packet with a block of file data. If the file is larger than one block, then the client computer sends an ACK (acknowledgement) packet, and the Server sends the next block of file data in response.

The CommuniGate Pro TFTP module supports relevant [Internet standards](#) (RFCs).

Configuring the TFTP module

Use a Web browser to configure the TFTP module. Open the Obscure page in the WebAdmin Settings section.

TFTP	
Log:	Problems listener
Default Site:	postmaster

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 Site

Since the TFTP module does not provide user authentication, you need to specify which Personal File Site is 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`.

Access to Account File Storage

The file name specified in the TFTP read 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.

The TFTP module tries to retrieve the specified files on behalf of the `tftpuser` Main Domain Account. By default, this Account does not exist, so the TFTP clients cannot retrieve anything from the `private` File Storage subdirectories.

To allow TFTP clients to access these subdirectories, create the `tftpuser` Account, and grant it the Unlimited File Storage Access right.

The addressed Account must have the WebSite Service enabled to allow TFTP clients to retrieve files from its File Storage.

Examples:

TFTP <i>filename</i> parameter	Addressed file
<code>file1.dat</code>	file1.dat in the Default Site
<code>dirA/file1.dat</code>	file1.dat in the dirA subdirectory of the Default Site
<code>/john/file1.dat</code> <code>~john/file1.dat</code>	file1.dat in the Account john Personal File Site
<code>/john/dirB/file1.dat</code> <code>~john/dirB/file1.dat</code>	file1.dat in the dirB subdirectory of the Account john Personal File Site
<code>/john@domain1.dom/dirB/file1.dat</code> <code>~john@domain1.dom/dirB/file1.dat</code>	file1.dat in the dirB subdirectory of



ACAP Module

The CommuniGate ACAP module implements an ACAP server for TCP/IP networks.

The ACAP protocol allows a client (mailer) application to connect to the Server computer and upload and download the application preferences, configuration settings and other datasets (such as personal address books).

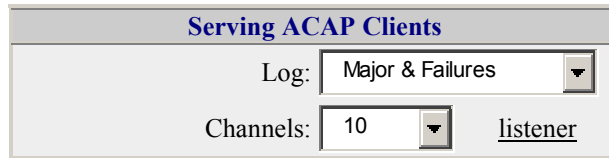
Application Configuration Access Protocol

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, and there are several mailer applications like Mulberry® that can work with the 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 a Web browser to configure the ACAP module. Open the Access page in the WebAdmin Settings section.



Serving ACAP Clients

Log: 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 mailers to establish a connection in the clear text mode and then turn it into a secure connection.



Services

The CommuniGate Pro Server allows users to use various applications and protocols. This section describes the protocols that do not deal directly with 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 [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.



HTTP Modules

The CommuniGate Pro HTTP Admin and User modules implement the Hypertext Transfer Protocol via TCP/IP networks.

The CommuniGate Pro Server uses the HTTP Admin module:

- to provide access to the [Server WebAdmin](#) (Administration) Interface pages.
- to provide access to [Domain WebAdmin](#) (Administration) Interface pages.

The CommuniGate Pro Server uses the HTTP User module:

- to implement the [WebUser Interface](#) to [user Accounts](#) and [Mailing List](#) archives.
- to provide access to [File Storage](#).
- to provide access to [Common Gateway Interface \(CGI\)](#) programs.

The CommuniGate Pro HTTP modules support various authentication methods, including the GSSAPI, [Kerberos](#), and [Certificate](#) methods implementing the *single sign-on* functionality.

Access to the Server WebAdmin Interface Pages

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 CommuniGatePro 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 name of the main server domain or its alias, or the IP address of the CommuniGate Pro Server.

Access to the Domain WebAdmin Interface Pages

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.

Server configuration errors can cut you off the Server WebAdmin Interface, if all your Server IP addresses and DNS names are assigned to secondary Domains. To access the Server WebAdmin 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 its IP addresses.

Other Domains can specify your Domain as their [Adminstrtor Domain](#). Your Domain WebAdmin Login page provides a list of those Domains, so you can open their WebAdmin Interfaces. Remember that you should login using your full Account name (*yourAccountName@yourDomainName*) when accessing other Domain WebAdmin pages.

Access to the WebUser Interface

CommuniGatePro 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 use 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 is available to the Account owner only, and it can be used to manage the File Storage.

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.

Access to Calendar and Tasks data

The CommuniGate Pro HTTP User module allows groupware applications (such as Apple iCal) to retrieve data from the Calendar and Task mailboxes in the iCalendar format. This operation is often called *subscribing* to Calendar data. The module also allows groupware applications to rewrite the mailbox content with data in the iCal-

endar format (to *publish* calendaring data).

The URL for Calendaring data is:

```
http://servername:port/CalendarData/mailboxName
```

where `port` is the WebUser port (8100 by default). If the `mailboxName` ends with the `ics` file extension, the server removes that extension.

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 WebUser 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 Task-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 mailbox, 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 mailbox, 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 mailbox, 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 exiting iCalendar items from the mailbox.

Configuring the HTTP modules

Use any Web Browser to connect to the Administration Port on your Server, and open the Access page in the Settings section.

Serving HTTP Admin Clients	
Log:	<input type="text" value="Failures"/>
Channels:	<input type="text" value="25"/> listener

Serving HTTP User Clients	
Log:	<input type="text" value="Low Level"/>
Channels:	<input type="text" value="250"/> listener

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.

Additional HTTP protocol options can be found on the WebUser page in the Setting realm of the WebAdmin Interface:

HTTP Options

☒ Advertise Basic AUTH

☐ Advertise Digest AUTH

☐ Advertise NTLM AUTH

☐ Advertise Negotiate AUTH

Request Size Limit: 3 Mbytes

☒ Support Keep-Alive

☒ Scan Large Requests

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.

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 the [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.

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.

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 built-in table:

File Extension	MIME type
html	text/html
txt	text/plain
gif	image/gif
jpg	image/jpeg

You can extend this table by specifying additional file name extensions and corresponding MIME Content-Types:

MIME types	
File Extension	MIME type
htm	text/html
zip	application/zip

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 a request is received on the WebAdmin port, the HTTP module should use the domain name or the IP address specified in the URL to decide which Domain Administration pages to display.

When a request is received on the WebUser port, the HTTP module should use the name specified in the URL to decide which Domain (its login page, mailing lists, Personal File Sites, 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` Personal File Site 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` Personal File Site 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 WebUser page in the Settings realm:

CGI Applications	
CGI Directory: <input type="text" value="myCGIs"/>	
File Extension	Program to Start
<input type="text" value="pl"/>	<input type="text" value="Perl.exe"/>
<input type="text"/>	<input type="text"/>
HTTP Header Fields	
<input type="text" value="X-Portal-De"/>	
<input type="text"/>	

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](#) module to access and modify mailbox data. To simplify these login operations, CGI programs can use the [WebUser Authentitcation](#) method.



LDAP Module

The CommuniGate 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.

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.

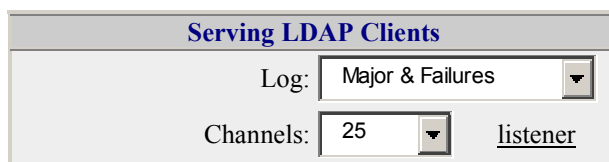
Very often LDAP services are used to look for names and E-mail addresses of Server users. But since the LDAP module provides access to the entire [Directory](#) tree, it can be used to work with any type of data placed into the CommuniGate Pro Directory. While the CommuniGate Pro Directory can be stored 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 a Web browser to configure the LDAP module. Open the Access page in the WebAdmin Settings section.



Serving LDAP Clients	
Log:	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 DN's 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-Based Domains](#) settings, and uses it as the Bind DN.

Sample:

If the Directory Integration settings are:

Base DN:	<code>o=myCompany</code>
Domain RDN attribute:	<code>cn</code>

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 (=) sign, or if it starts with the `mail=` symbols and does not contain any other equals (=) signs.

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 theou=human_resources,o=myCompany Directory record
user@domain.com	the user@domain.com Account password
mail=user@domain.com	the user@domain.com Account password
uid=user,cn=domain.dom,o=myCompany (LDAP Provisioning is on)	the user@domain.com 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

Very often the LDAP services are 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-Based Domains](#) 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.

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 `@` sign to build the `uid-Value@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 `@` sign, only the part of the string before that sign is used.

These two features can be enabled or disabled using the Domain Integration page in the Domains realm of the

WebAdmin Interface:

LDAP Attribute Processing	
<input checked="" type="checkbox"/>	Substitute mail with uid in conditions
<input checked="" type="checkbox"/>	Compose mail using uid
<input checked="" type="checkbox"/>	Ignore objectCategory filters

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.



PWD Module

The CommuniGate PWD module implements a poppwd server for TCP/IP networks.

The poppwd protocol allows a client (mailer) 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 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 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

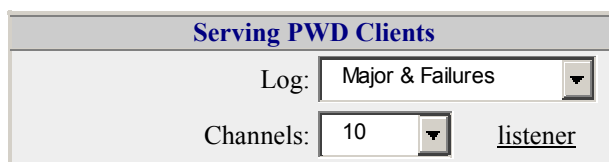
password if this option is disabled.

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 configure the PWD module. Open the Access page in the WebAdmin Settings section.



Serving PWD Clients	
Log:	Major & Failures ▼
Channels:	10 ▼ 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. Since the `popppwd` sends passwords in the clear text format, setting the Log to these setting 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 all PWD connections that mail clients establish in order to updates password data. 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](#) chapter for the details.



RADIUS Module

The CommuniGate Pro Server supports RADIUS authentication for various NAS (Network Access Servers). 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
- 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.

By default the CommuniGate Pro RADIUS server is not activated.

Configuring the RADIUS Module

To configure the RADIUS module, use the WebAdmin Interface. Open the Obscure page in the Settings section and find the RADIUS panel:

RADIUS	
Log:	Problems <input type="button" value="v"/> listener
Password:	*****
Channels:	3 <input type="button" value="v"/> <input type="checkbox"/> 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).

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 request 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.

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:

<input checked="" type="checkbox"/> External RADIUS			
Log:	Major & Failures	Program Path:	helpers/RADIUSpolicy
Time-out:	disabled	Auto-Restart:	15 seconds

See the [Helper Programs](#) 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 [Helpers](#) section.

Sample External RADIUS programs and scripts can be found at the <http://www.stalker.com/CGRADIUS/> 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 attributes is encoded with the same, the numeric attribute type, the equal (=) sign, 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.



SNMP Agent

The CommuniGate Pro Server internal information can 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.

By default, the CommuniGate Pro SNMP agent is not activated.

Configuring SNMP Agent

To configure the SNMP agent, use the WebAdmin Interface. Open the Obscure page in the Settings section and find the SNMP Agent panel:

SNMP Agent	
Log:	Problems listener
Password:	*****
Trap Password:	

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.

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.

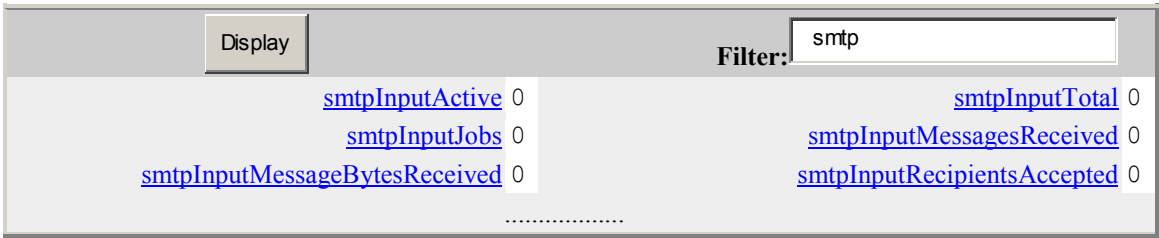
Different versions of the CommuniGate Pro software support different sets of internal objects, and CommuniGatePro Server generates its MIB by a user request, presenting the most current information.

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.

Monitoring SNMP elements via Web

The SNMP module allows a Server Administrator to monitor the server internal parameters via Web. Open the SNMP page in the Monitors section of the [Web Admin](#) Interface. You need to have the [Can Monitor Server](#) [Server Access Right](#) to open this page.



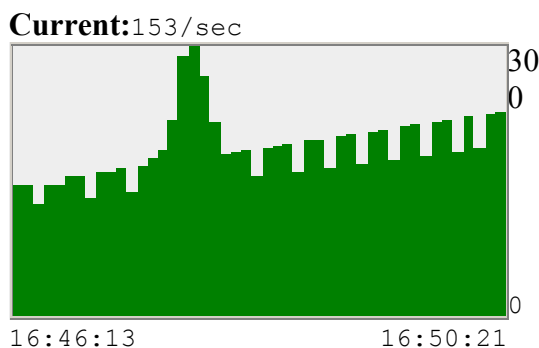
The screenshot shows a web interface for monitoring SNMP elements. At the top, there is a 'Display' button and a 'Filter:' field containing the text 'smtp'. Below this is a table with two columns of element names and their current values. The elements listed are smtpInputActive, smtpInputJobs, smtpInputMessageBytesReceived, smtpInputTotal, smtpInputMessagesReceived, and smtpInputRecipientsAccepted. All values are currently 0. The table is followed by a line of dots.

smtpInputActive	0	smtpInputTotal	0
smtpInputJobs	0	smtpInputMessagesReceived	0
smtpInputMessageBytesReceived	0	smtpInputRecipientsAccepted	0
.....			

The page contains the list of the SNMP (MIB) 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 SNMP elements by name.

The Element Monitor page contains a histogram allowing you to monitor how the element value changes over time:



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 SNMP Web Monitor system.

Monitoring SNMP elements via CLI/API

The [CLI/API](#) GetSNMPElement command allows a Server Administrator to monitor the server internal parameters via the CLI/API interface and via various CLI "wrappers".

Sending SNMP Traps

The SNMP module can send Traps on certain Events. See the [Events](#) section of the manual 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.



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.

Configuring the BSD Log Module

To configure the BSD Log module, use the WebAdmin Interface. Open the Obscure page in the Settings section and find the BSD Log panel:

BSD Log	
Log:	<div>Problems ▼</div> <div>listener</div>
<input type="checkbox"/>	Use OS <code>syslog</code>

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 BSD Log module stores all received records via the Server OS syslog service.



Directory

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 (,) sign.

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:

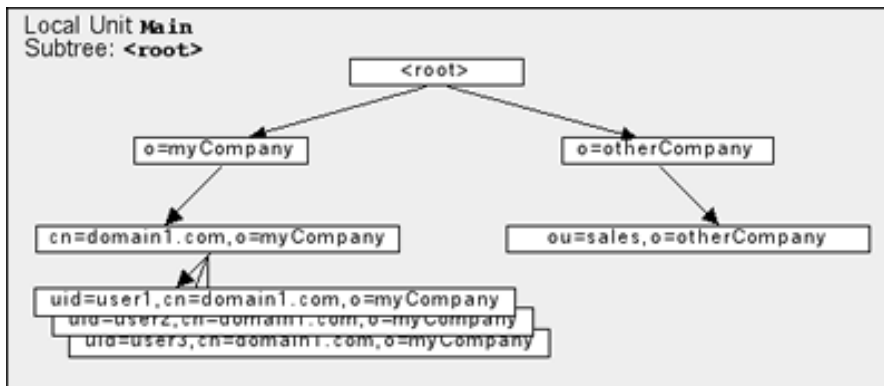
- *local* units - sets of files managed with the CommuniGate Pro File Directory Manager. These files contain directory records, replication information, and subtree schemas.
- *remote* units - descriptors managed with the CommuniGate Pro LDAP Directory Manager. These descriptors contain the information 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.

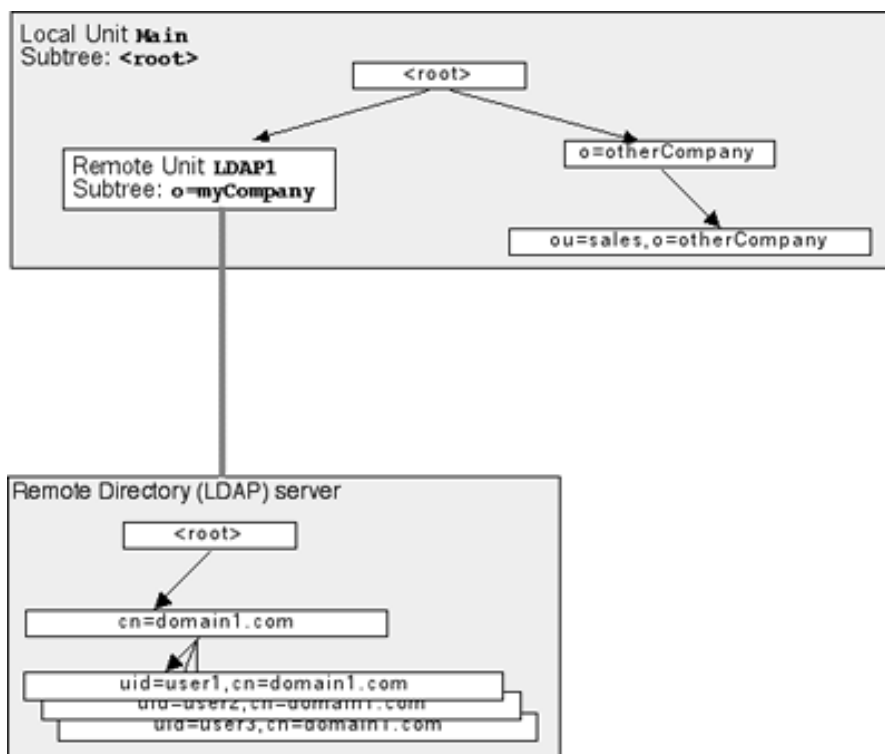
Initially, the CommuniGate Pro server creates one Local Storage Unit `Main` that contains the entire directory.

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 left frame of the CommuniGate Pro Server WebAdmin Interface. The Directory Storage Units page opens. You need to have the Directory access right to open this page.

Display

Filter:

2 of 2 Storage Units selected

Subtree	Unit Name
<root>	Main
o=node6	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 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

Log:

Major & Failures

☒ Enforce Schema

Search Result Limit:

30

records

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.

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 Unit

Click a Remote Storage Unit name on the Directory WebAdmin page to open the unit Settings page:

Settings

Log:	Major & Failures ▼
LDAP Server Name:	node6.stalker.com
Security:	Disabled ▼
Server Subtree:	
BIND DN:	cn=admin,o=Stalker Labs,c=US
BIND Password:	*****
Protocol Version:	LDAPv2 ▼
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.

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 [Web User Interface](#) 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

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	Rights	
	BrotherWa	*	brother	allow	specifications	Down
Up	ChildSearc	*	child	allow	specifications	Down
Up	DirSearch	o=MyCompany	*	allow	specifications	
				allow		

Each Access Right record has:

- a name.
- a target: the DN the record applies to; wildcard characters ("*") can be used in Target strings.
- a Bind DN: the client binding DN the record applies to.
- 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:

`brother`

the Access Right record is applied if the Bind DN and the Target DN has the same parent DN. For example, the `uid=someuser, cn=domain1.com` and `uid=otheruser, cn=domain1.com` DN's are "brothers". 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` DN's.

`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.

Access Right Specifications

To specify Directory Access Rights, open the Access Rights page and click the "specifications" link to open the Access Right record:

Entry-Level	
<input type="checkbox"/> delete entry	<input type="checkbox"/> create entry

These options specify if clients with the given Bind DN can create or delete records with the given Target DN.

Readable Attributes	
<div></div>	

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 ("*") sign.

Searchable Attributes	

This field lists the attributes that clients with the given Bind DN can use in filters when searching Target DN subtrees.

Modifiable Attributes	

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

`brother`

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 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 page and click the Browser link to open the Directory Browser page.

The Browser page includes the DN field:

Distinguished Name		
Up	<input type="text" value="o=Stalker Labs,c=US"/>	Go


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	"\"subsidiary for Stalker Software, Inc.\""
o	"Stalker Labs"
objectclass	organization
seealso	"o=Stalker Software, 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	
Display	1000 
Filter:	<input type="text"/>
RDN	objectClass
cn=aaa.com	CommuniGateDomain
cn=bbb.com	CommuniGateDomain

Use the pop-up menu to limit the number of records displayed on the subtree panel.

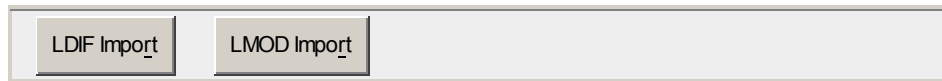
To search for specific records, enter an LDAP filter string (in the RFC 2254 format) into the Filter field and click the Display button.

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 "replug" formats:



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 "replug" file on your workstation. Click the LMOD Import button to apply all modifications from the selected file.



Directory Schema

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

Add Attribute

Name:

ObjectID:

Name	ObjectID	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:

Add Class

Name:

ObjectID:

Parent:

top

Name	ObjectID	Parent Class	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	1	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:

Add Required Attribute

Add Optional Attribute

objectClass

Required Attributes	Optional Attributes
cn, sn	description, telephoneNumber, facsimileTelephoneNumber, userPassword
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 Integration

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 on that page.

Central Directory 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 updated, 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 Renaming table to specify the name translation rules:

Attributes Renaming	
Name in CommuniGate	Name in Directory
<input type="text" value="city"/>	<input type="text" value="l"/>
<input type="text" value="Password"/>	<input type="text" value="userPassword"/>
<input type="text" value="RealName"/>	<input type="text" value="cn"/>
<input type="text" value="surname"/>	<input type="text" value="sn"/>
<input type="text"/>	<input type="text"/>

Note: The Attributes Renaming 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 renaming 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:

Domains Subtree	
Base DN	<input type="text" value="o=Our Company"/> <input type="button" value="Create It"/>
Domain RDN attribute	<input type="text" value="cn"/>
Domain objectClass	<input type="text" value="CommuniGateDomain"/>
UID subtree	<input type="text"/>

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 DN:

o=domain name,base DN

Note: If you specify the string dc as the Domain RDN attribute, then the DN for a CommuniGatePro 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 domains, 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, groups, lists, forwarders) 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:

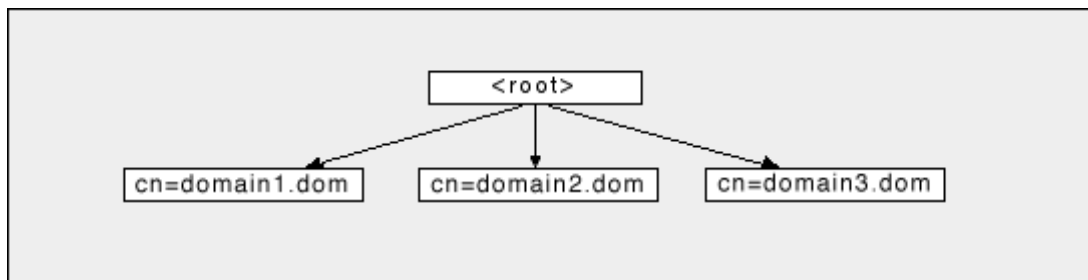
```
uid=user1,ou=People,cn=domain1.dom,o=mycompany
```

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:



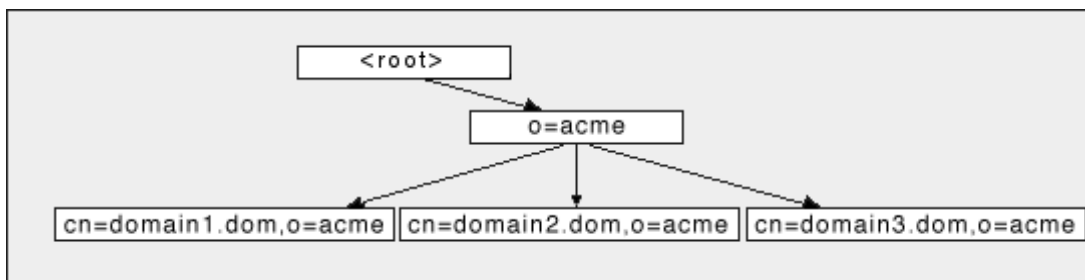
To search for accounts in these domain subtrees, LDAP clients should have the string

`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:



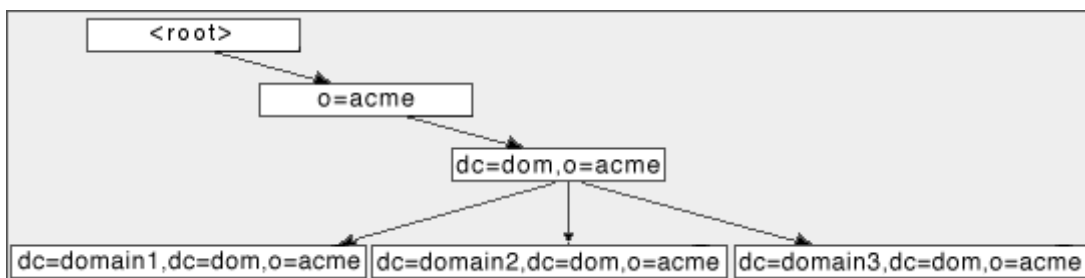
To search for accounts in these domain subtrees, LDAP clients should have the string

`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:

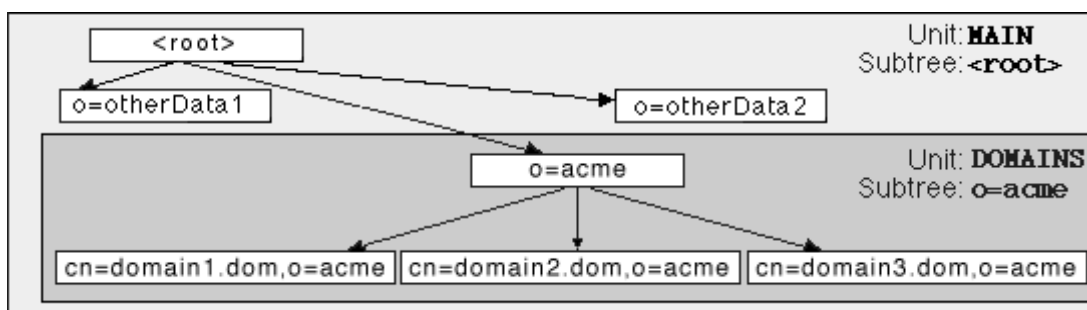


To search for accounts in these domain subtrees, LDAP clients should have the string

`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
<input type="text" value="surname"/>	<input type="text" value="telephoneNumber"/>
<input type="text" value="city"/>	<input type="text"/>
<input type="text"/>	

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 an empty 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:	<input type="text" value="John R. Smith"/>
surname:	<input type="text" value="Smith"/>
city:	<input type="text" value="Timbuktu"/>
CommuniGate Password:	<input type="password" value="*****"/>
telephoneNumber:	<input type="text" value="800 262 4722"/>

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`.

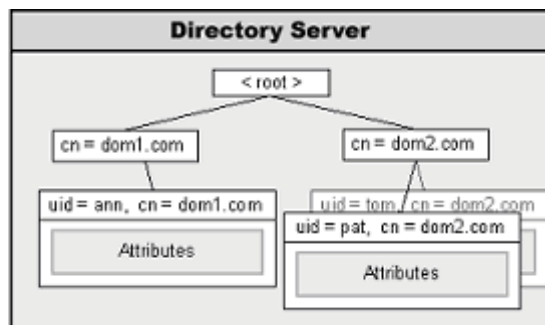
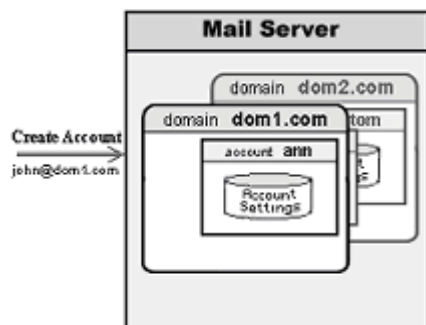
When the [Directory Integration](#) Domain Setting is set to `Keep In Sync`:

- 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 DN's are renamed when Domain objects are renamed.
- Directory records are updated when Domain Accounts settings are updated.

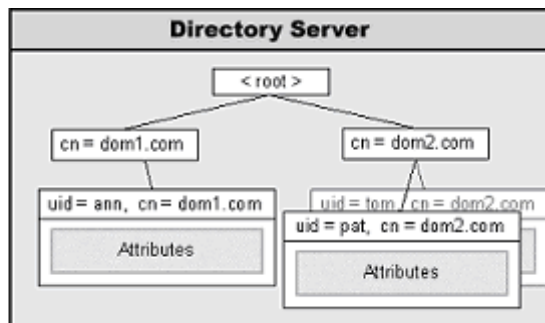
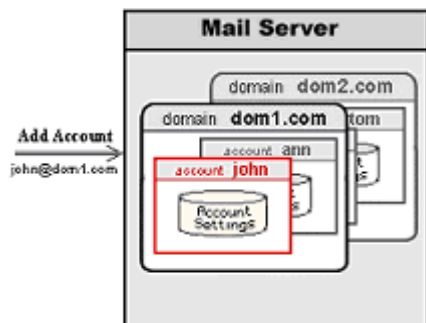
None of these actions takes place when the Domain Directory Integration settings is set to `Disabled`.

The following diagram illustrates what happens when the `dom1.com` Domain has the Directory Integration option set to `Keep In Sync`, and an account is created in that domain:

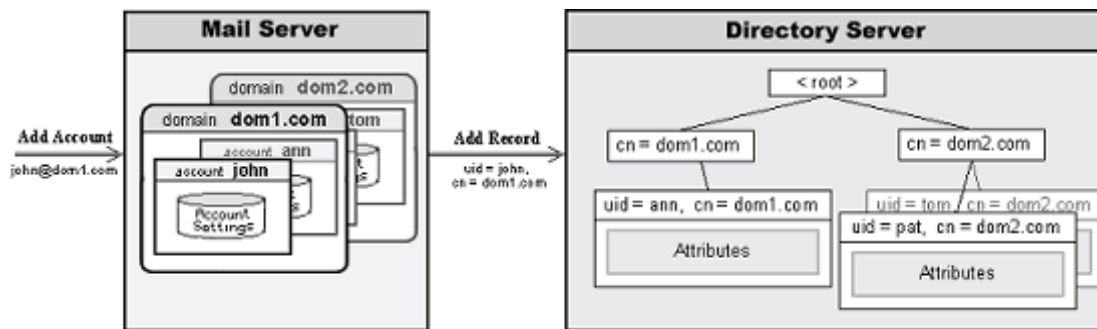
Step 1.



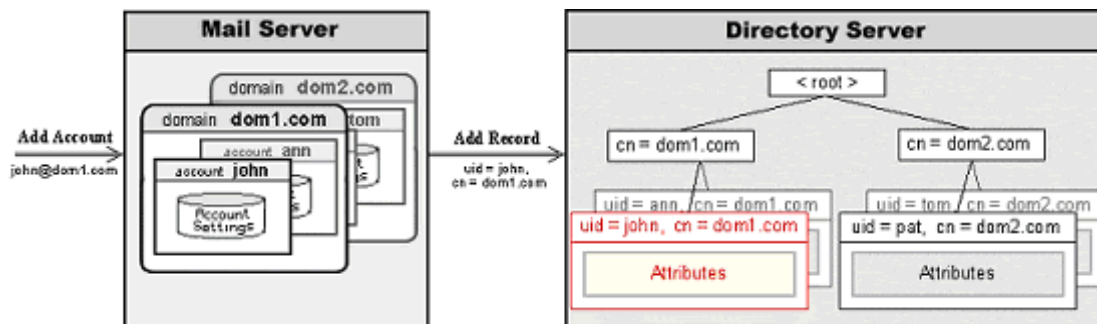
Step 2.



Step 3.



Step 4.



In this example:

- The WebAdmin or CLI interface is used to create the john Account in the dom1 . com 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	
Copy into Account records:	<input checked="" type="checkbox"/> Passwords <input type="checkbox"/> Standard Settings

Copy Password

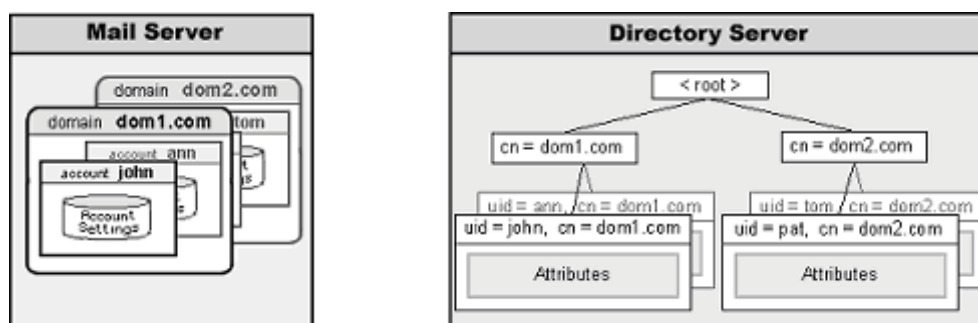
If this option is selected, the Directory records for Regular Domain Accounts will contain the Account Password attribute (usually it is renamed into the `userPassword` attribute).

Copy Standard Settings

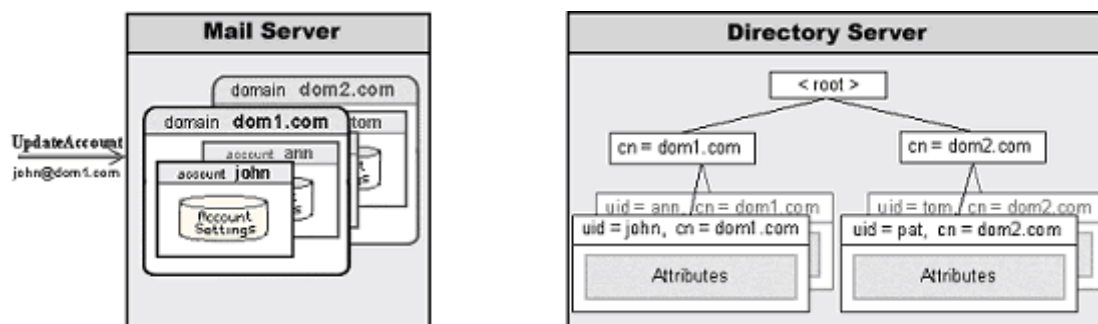
If this option is selected, the Directory records for Regular Domain Accounts will contain all CommuniGate Pro standard Settings for those Accounts (excluding the `RealName` and `userCertificate` settings that are always being stored and the Password setting that it controlled using a separate option).

The following diagram illustrates what happens when the dom1.com Domain has the Directory Integration option set to Keep In Sync, and Domain Account settings are updated:

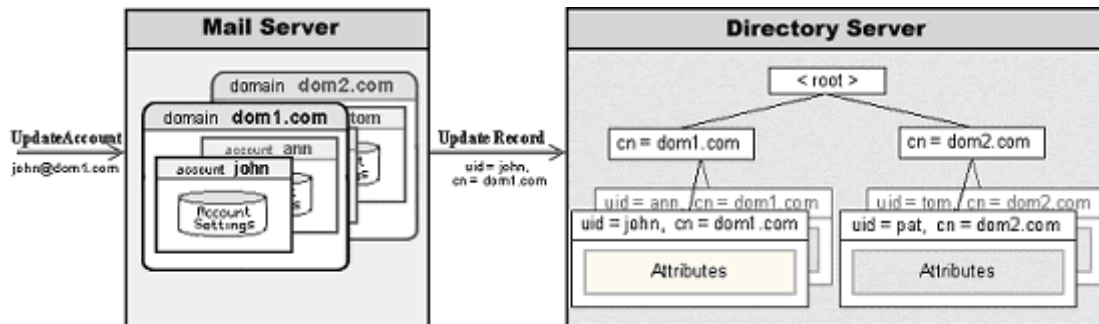
Step 1.



Step 2.



Step 3.



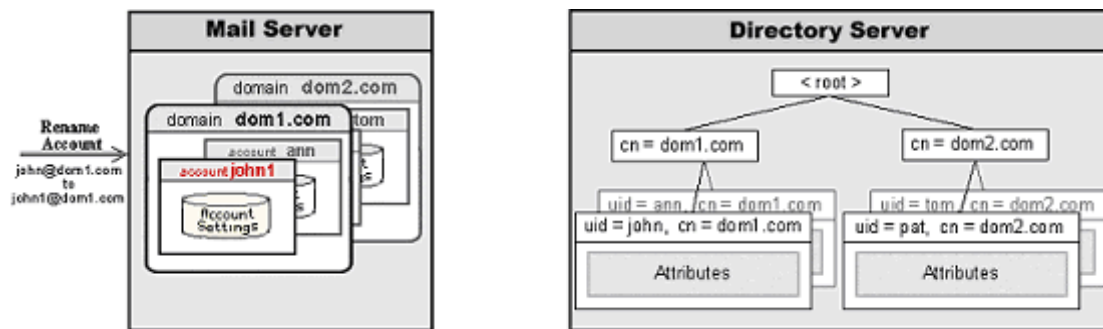
In this example:

- 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.com 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.

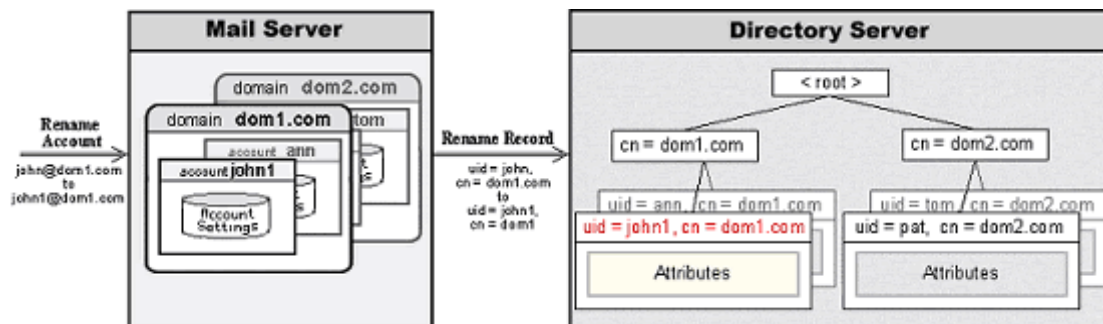
Note: It is important to understand that Directory Integration for Regular Domains is a one-way relationship: if you change attributes of Account records in the Directory (using any LDAP utility), the actual Account Settings will not be modified - CommuniGate Pro always uses data in the settings files, 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.com Domain has the Directory Integration option set to Keep In Sync, and a Domain Account is renamed:

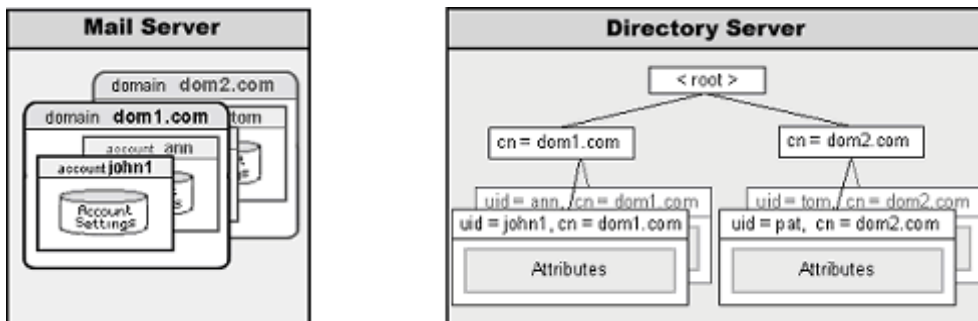
Step 1.



Step 2.



Step 3.

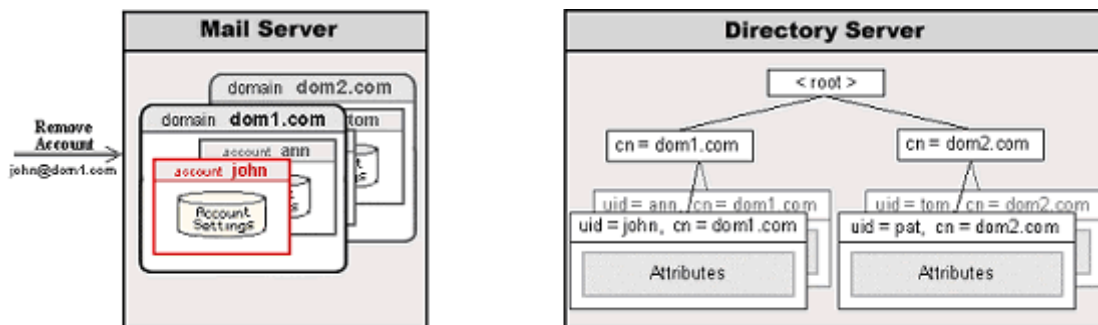


In this example:

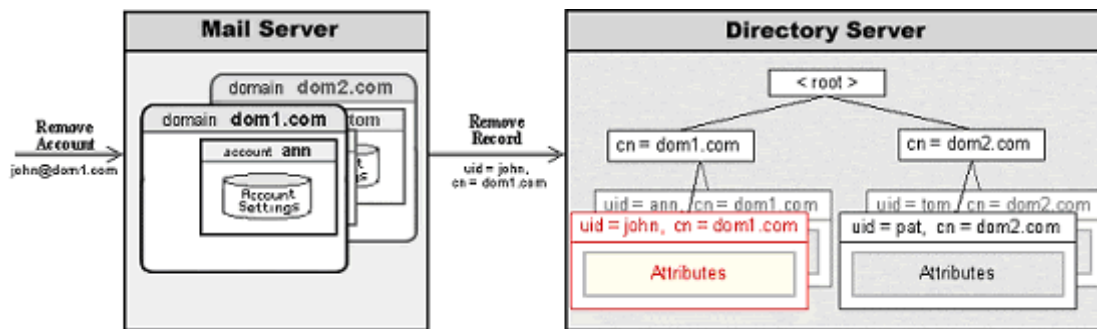
- The RenameAccount operation is initiated with a WebAdmin or CLI Interface command.
- The Account john and its files are renamed.
- If the dom1.com Domain Directory Integration setting is set to Keep In Sync, the Account Manager executes the RenameRecord (modifyDN) Directory operation to rename the Account record in the Directory.

The following diagram illustrates what happens when the dom1.com Domain has the Directory Integration option set to Keep In Sync, and a Domain Account is removed:

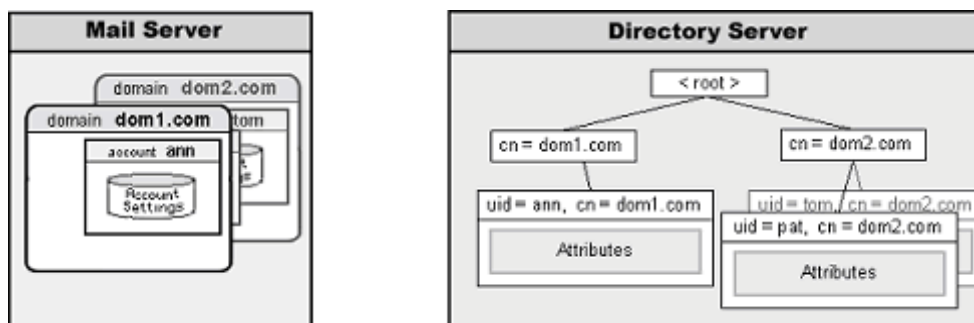
Step 1.



Step 2.



Step 3.



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.com 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

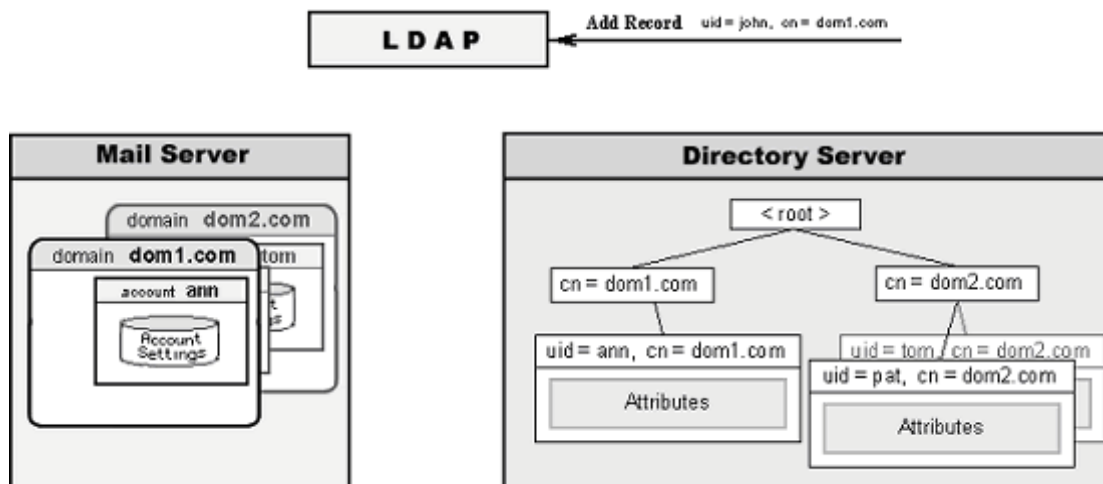
CommuniGate Pro allows you to use the LDAP protocol to create, update, rename, and remove Accounts:



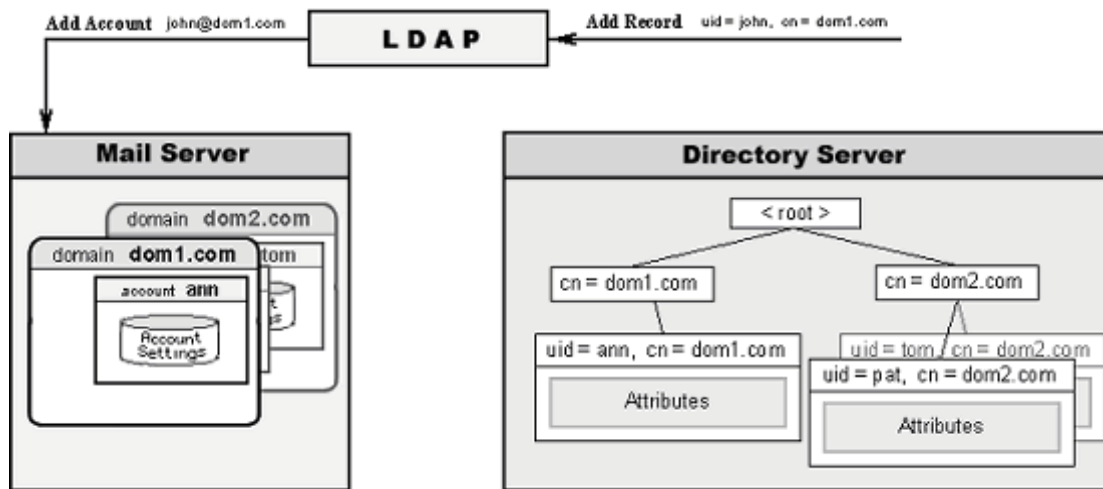
When this option is enabled, the LDAP module checks the names (DNs) specified in the update operations. If the DN looks like a DN of a CommuniGate Pro Account, the LDAP module does not perform the requested operation with the Directory. Instead, it executes the **CreateAccount**, **UpdateAccount**, **RenameAccount**, or **RemoveAccount** operations for the specified Account and Domain.

The diagram below illustrates how the LDAP **AddRecord** operation works in this case:

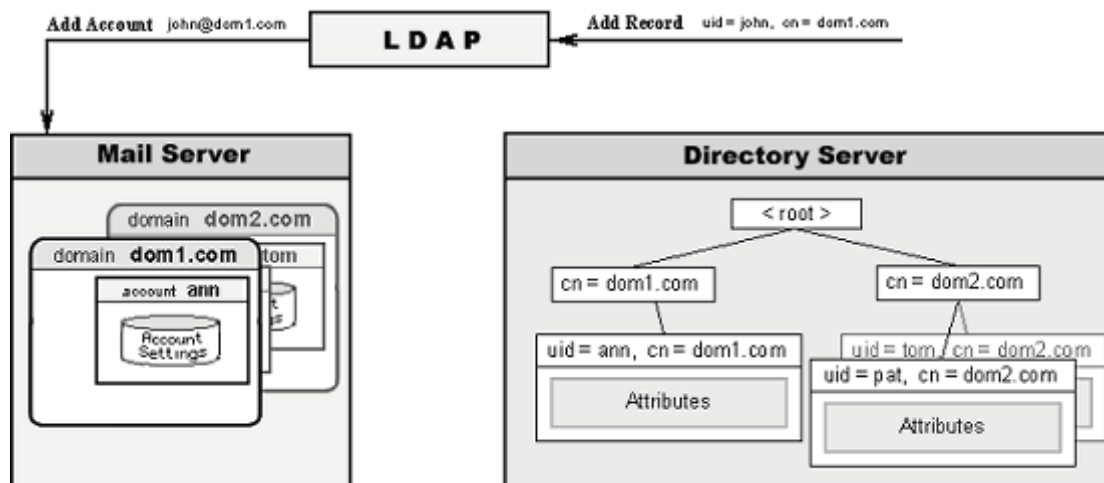
Step 1.



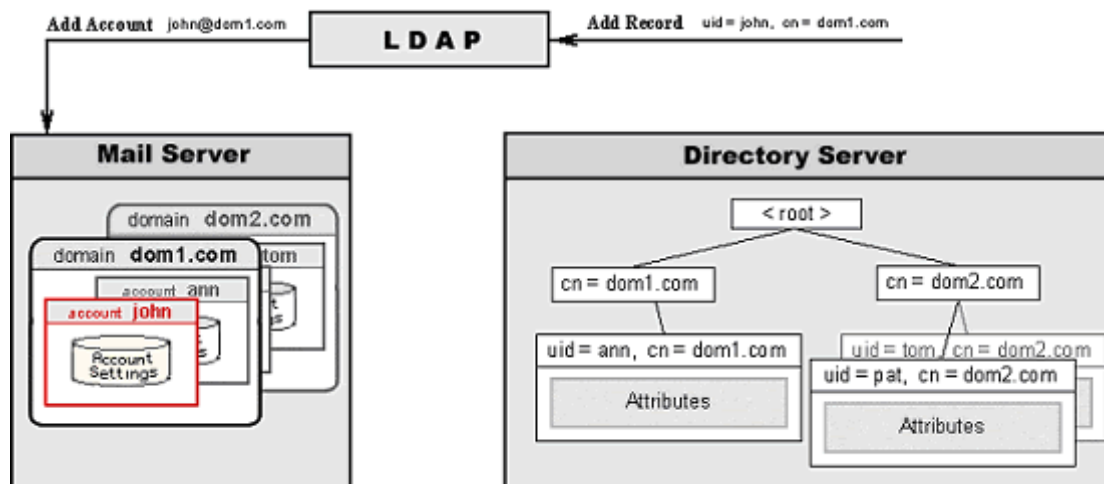
Step 2.



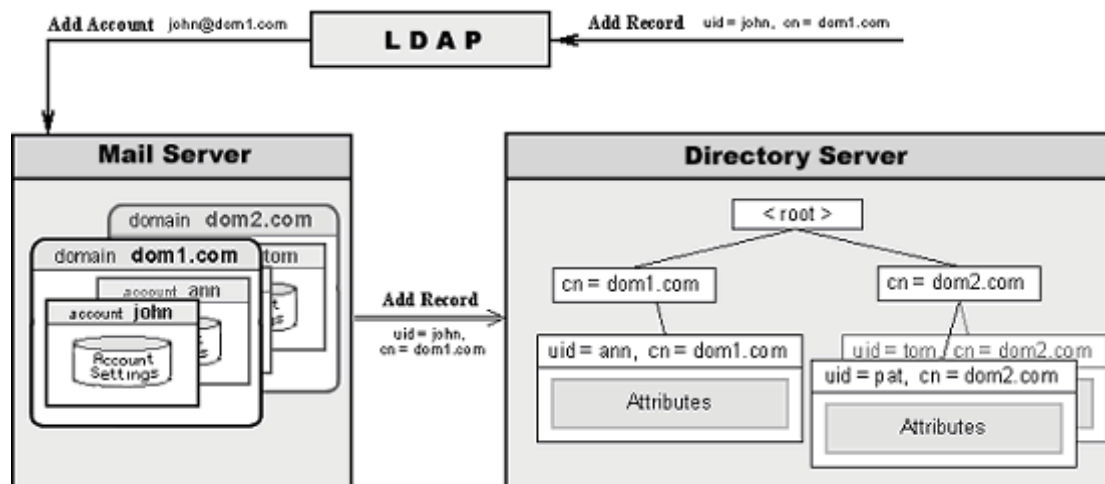
Step 3.



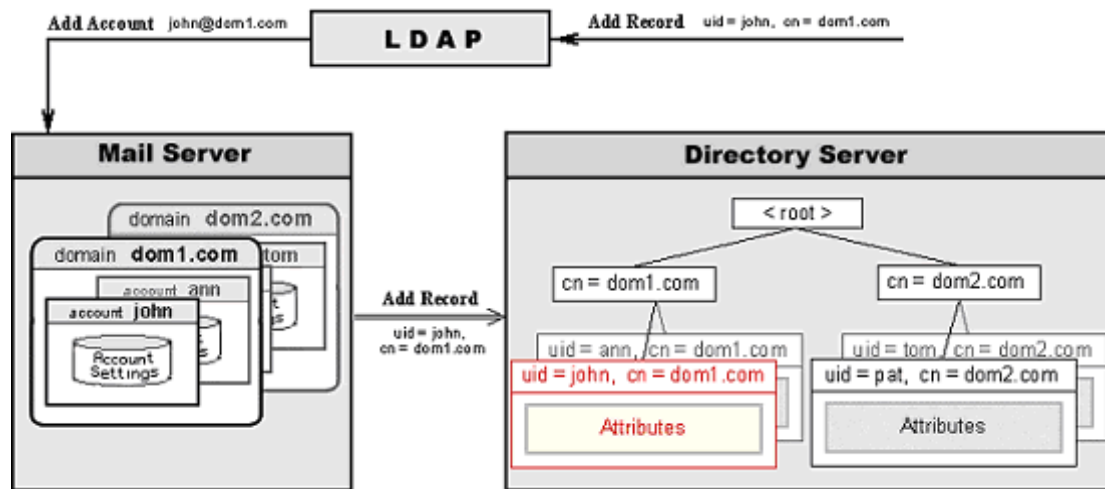
Step 4.



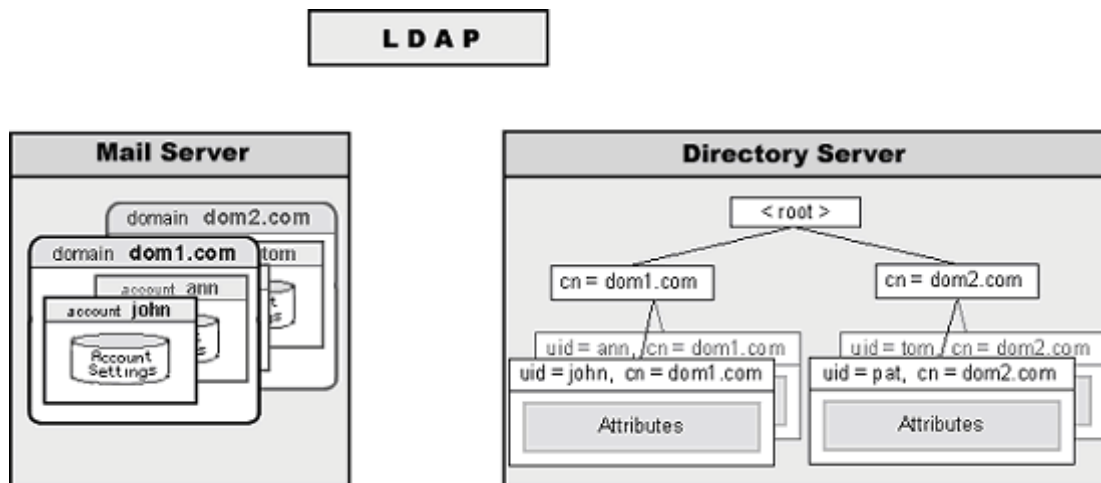
Step 5.



Step 6.



Step 7.



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.com` 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.com`, and the CommuniGate Pro Domain `dom1.com` does exist.
- Instead of performing the requested AddRecord operation with the Directory, the LDAP module executes the `CreateAccount(john)` operation in the `dom1.com` 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.com` 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

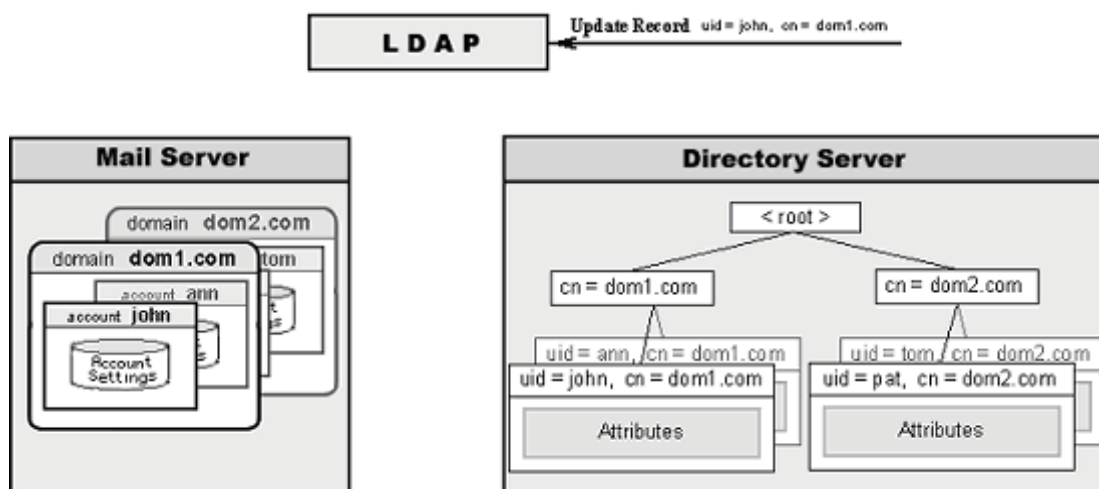
stored in the Account settings as the Account RealName setting. When the Account Manager adds a record to the Directory, it converts the RealName Account setting back into the cn record attribute.

Note: all LDAP AddRecord request attributes will be stored as the Account Settings. 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.

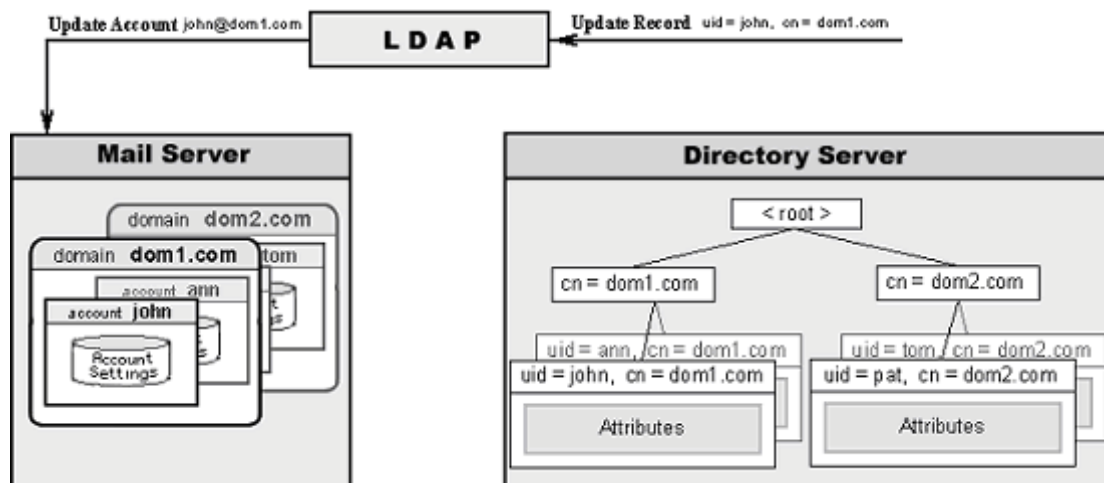
Note: the LDAP Provisioning feature detects the `unixPassword` attributes and converts them into Password settings after adding a leading 0x02 byte. See the [Account Import](#) section for the details.

The following diagram illustrates how the LDAP ModifyRecord operation can be used to modify Account Settings:

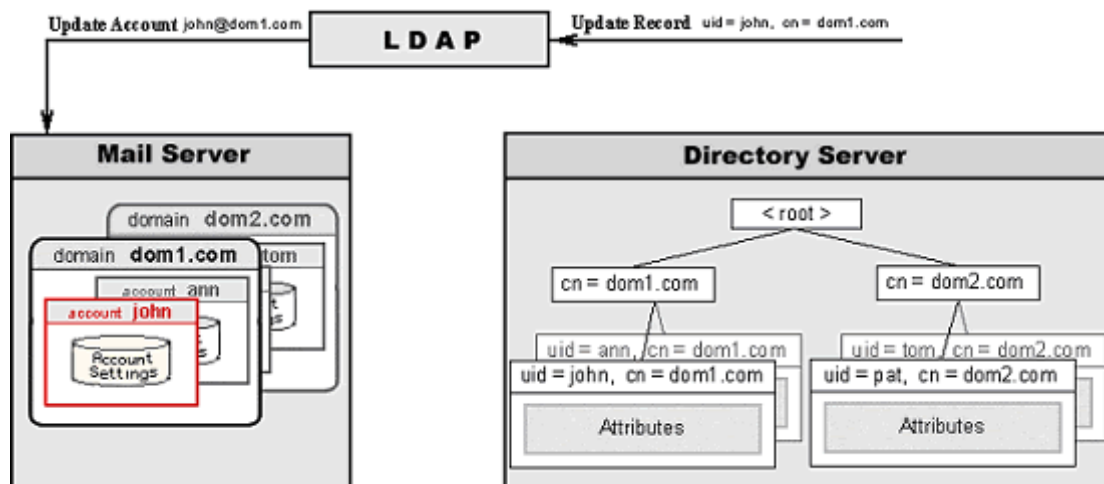
Step 1.



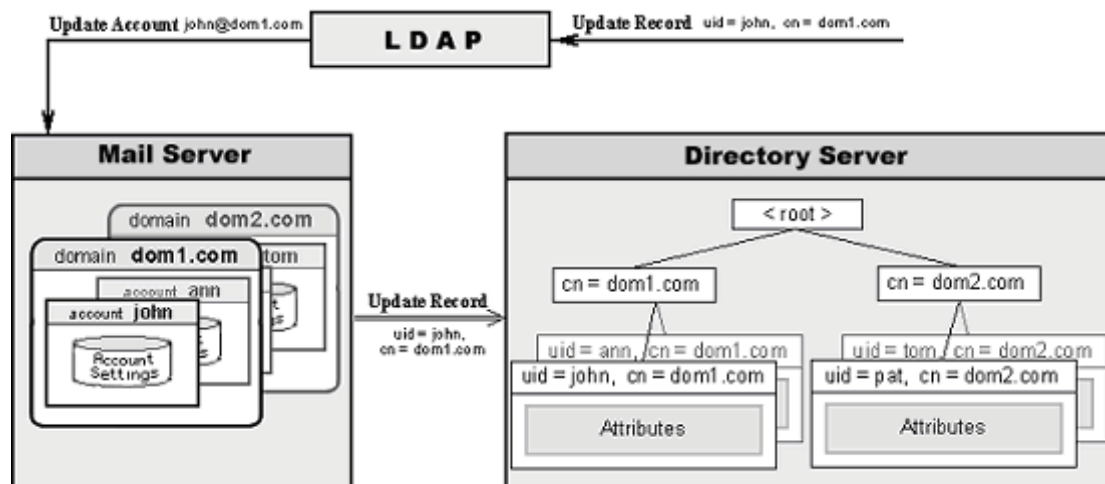
Step 2.



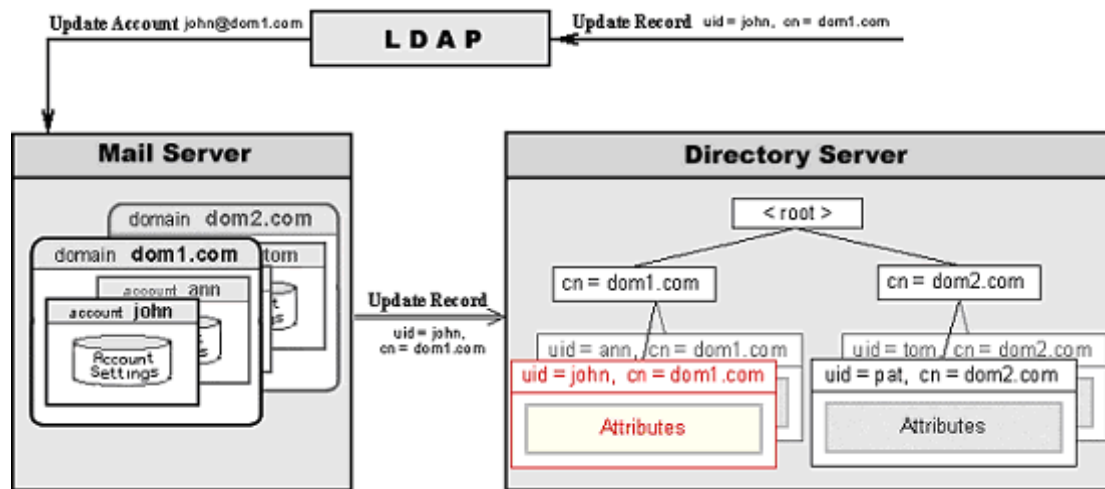
Step 3.



Step 4.



Step 5.

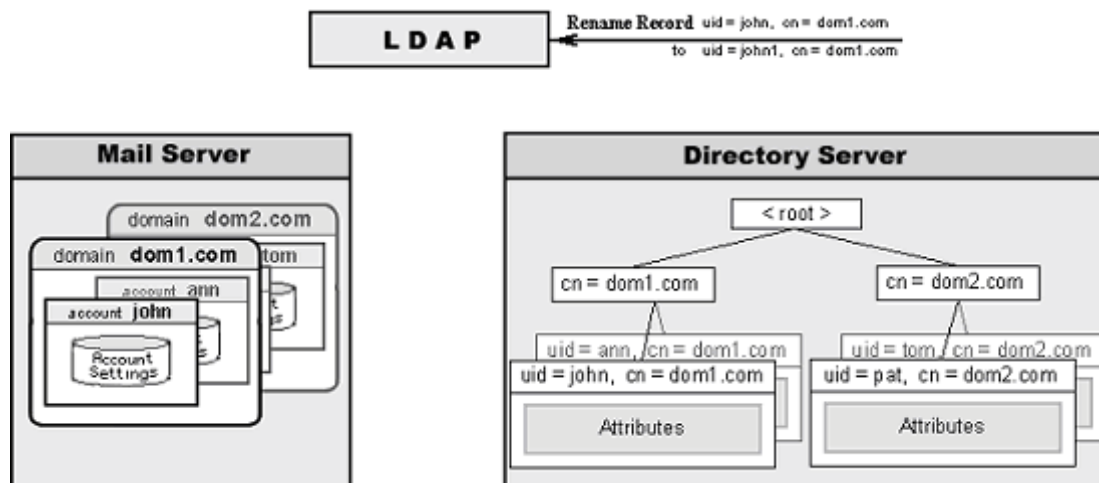


In this example:

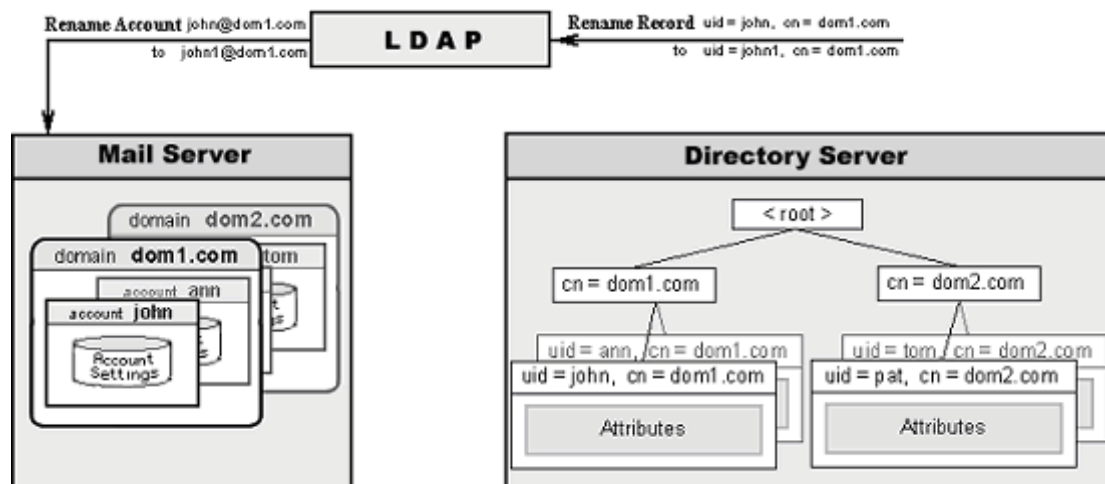
- The LDAP module received a ModifyRecord request from an LDAP client. The client asks the LDAP module to update a record with the `uid=john, cn=dom1.com` DN.
- The LDAP module checks the DN and sees that it looks like the record DN for the CommuniGate Pro Account `john` in the CommuniGate Pro Domain `dom1.dom`, and the CommuniGate Pro Domain `dom1.com` 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.com` Domain Directory Integration setting is set to Keep In Sync, the Account Manager executes the ModifyRecord Directory operation to create a record in the Directory.

The following diagram illustrates how the LDAP ModifyDN operation can be used to rename Accounts:

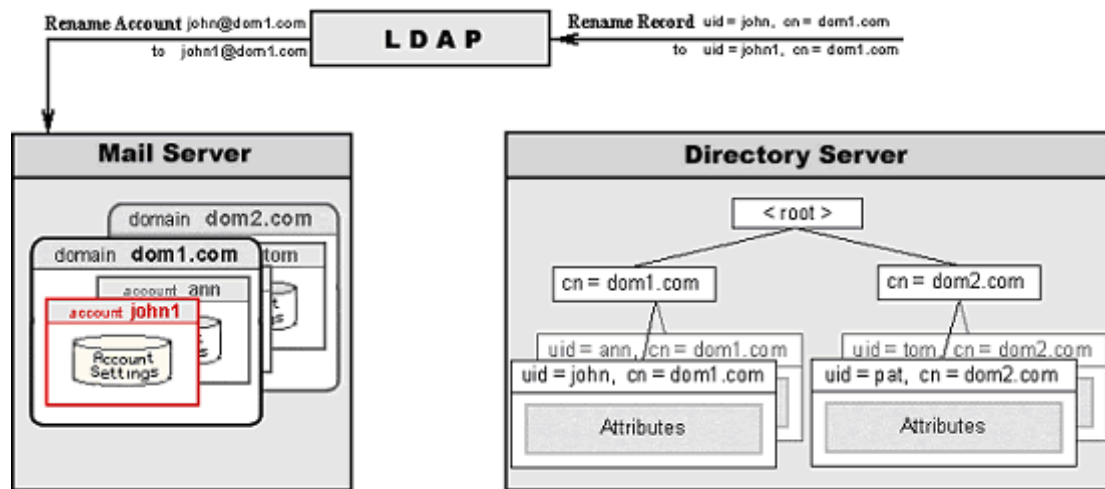
Step 1.



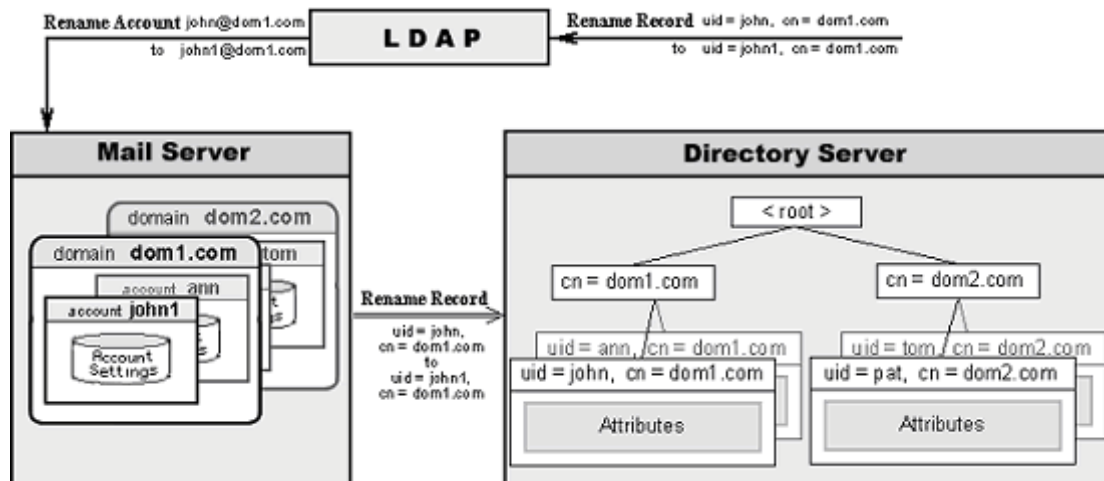
Step 2.



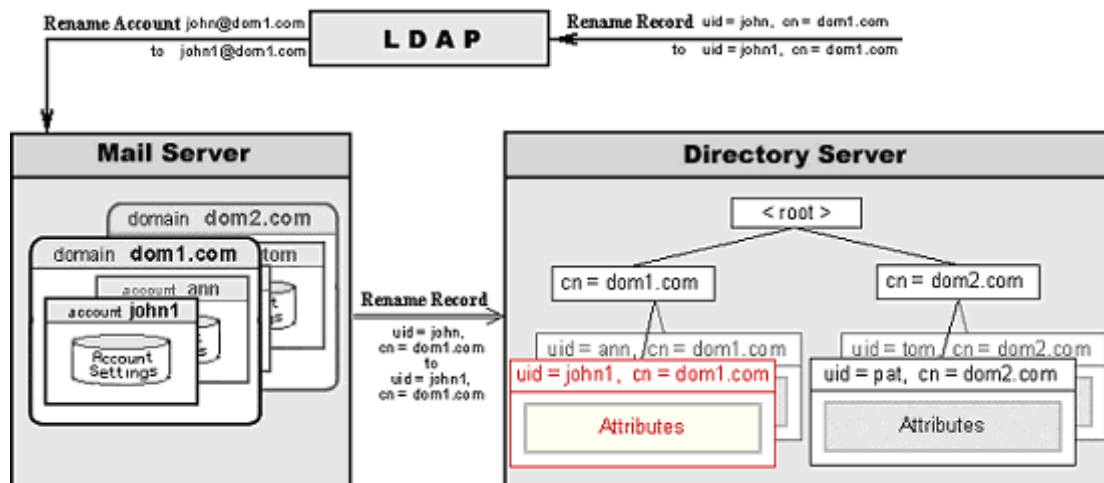
Step 3.



Step 4.



Step 5.

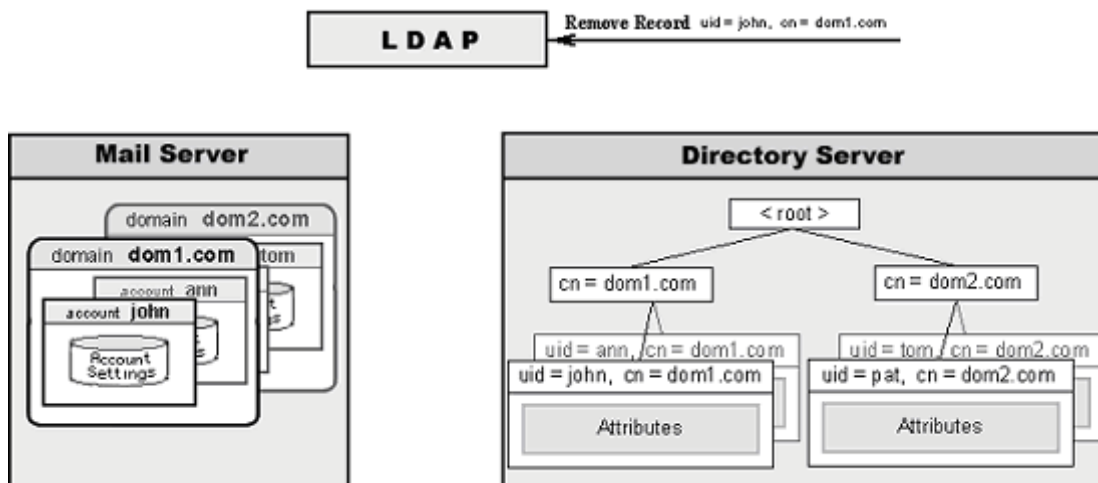


In this example:

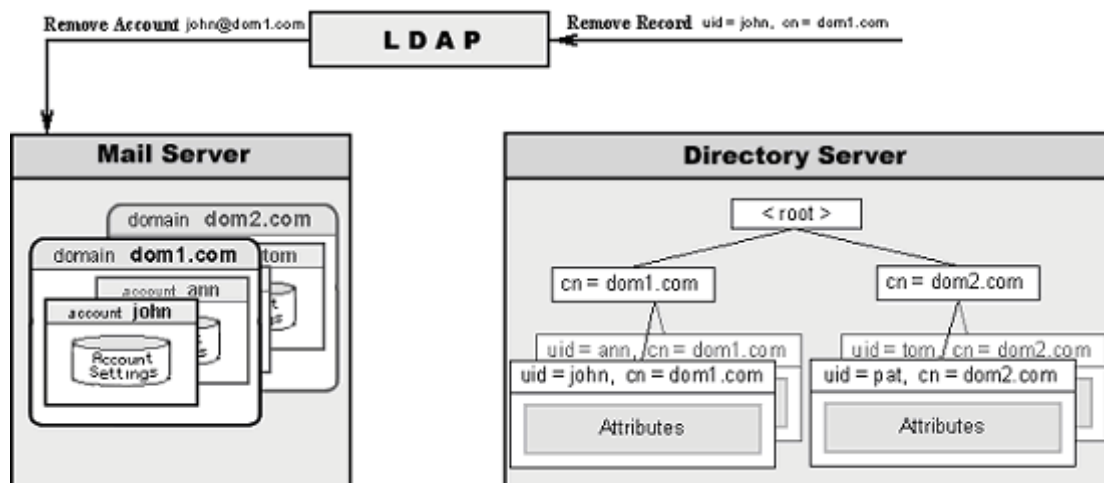
- The LDAP module received a ModifyDN request from an LDAP client. The client asks the LDAP module to change the `uid=john, cn=dom1.com` record DN.
- The LDAP module checks the DN and sees that it looks like the record DN for the CommuniGate Pro Account `john` in the CommuniGate Pro Domain `dom1.com`, and the CommuniGate Pro Domain `dom1.com` 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.com` Domain.
- The Account `john` is renamed into `john1`.
- If the `dom1.com` 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:

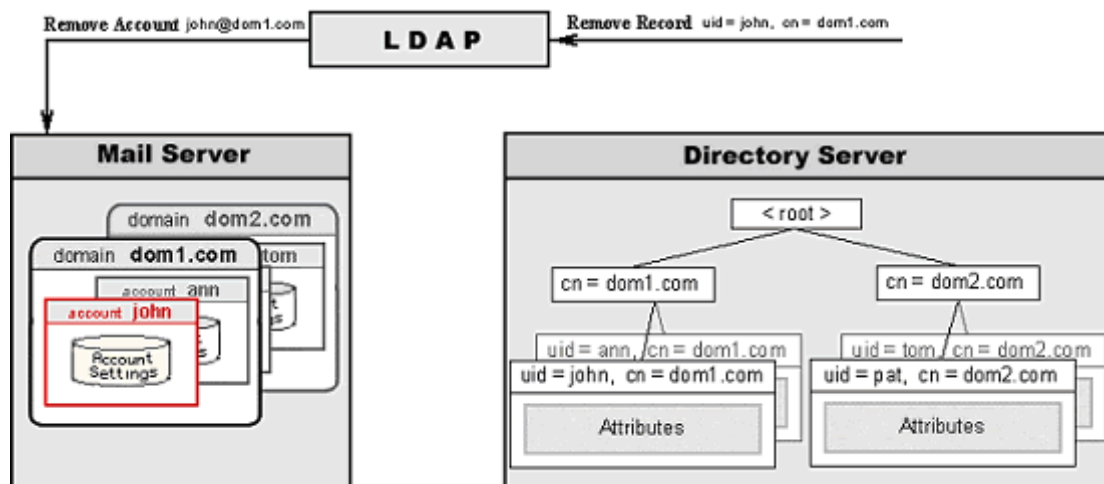
Step 1.



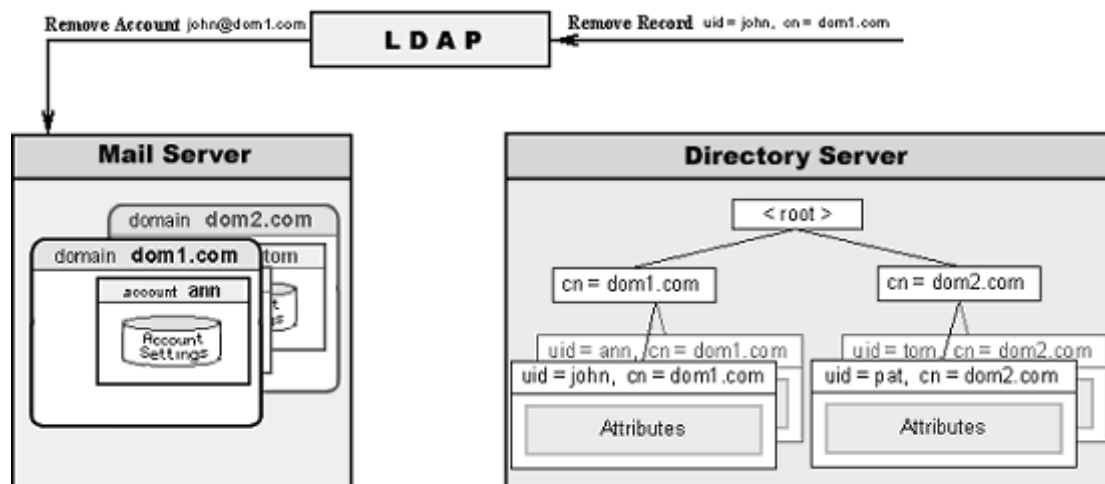
Step 2.



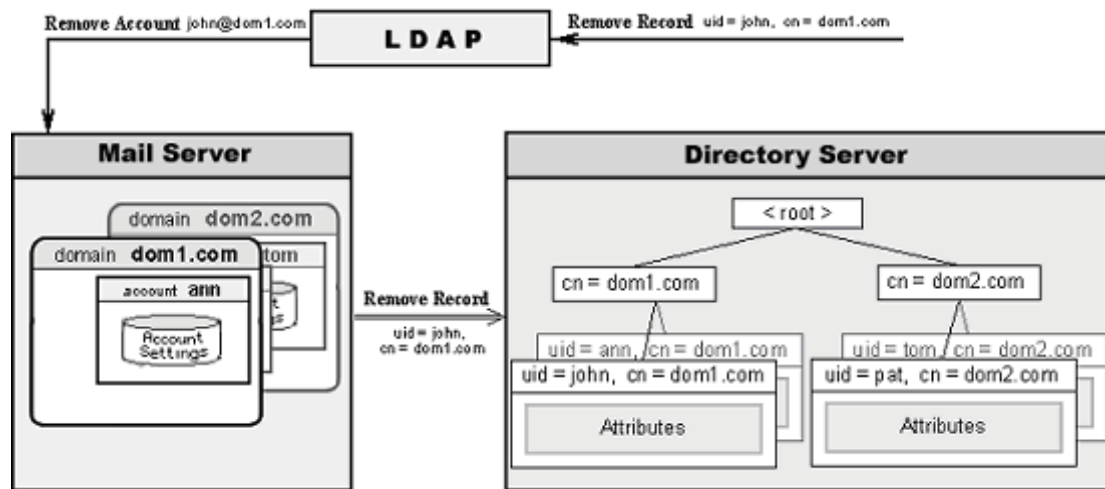
Step 3.



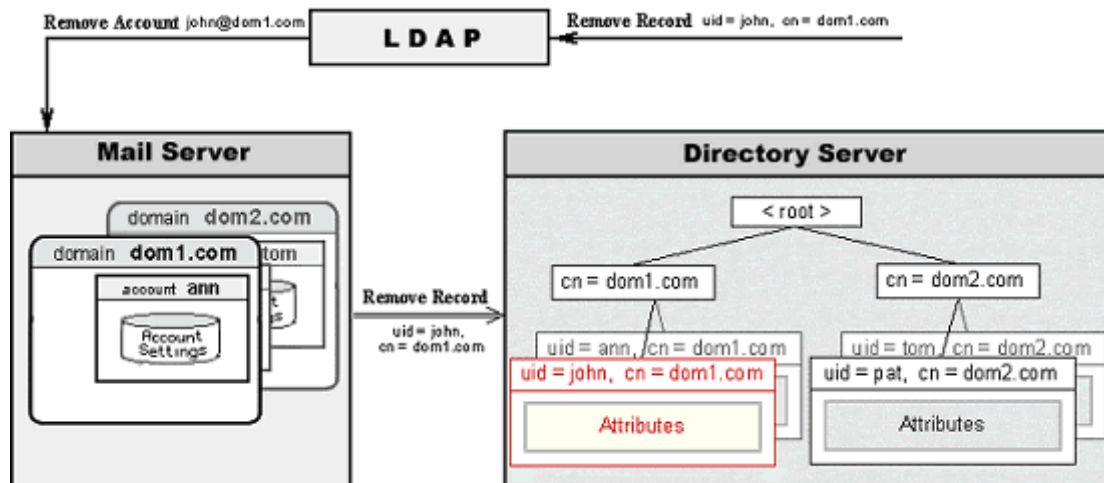
Step 4.



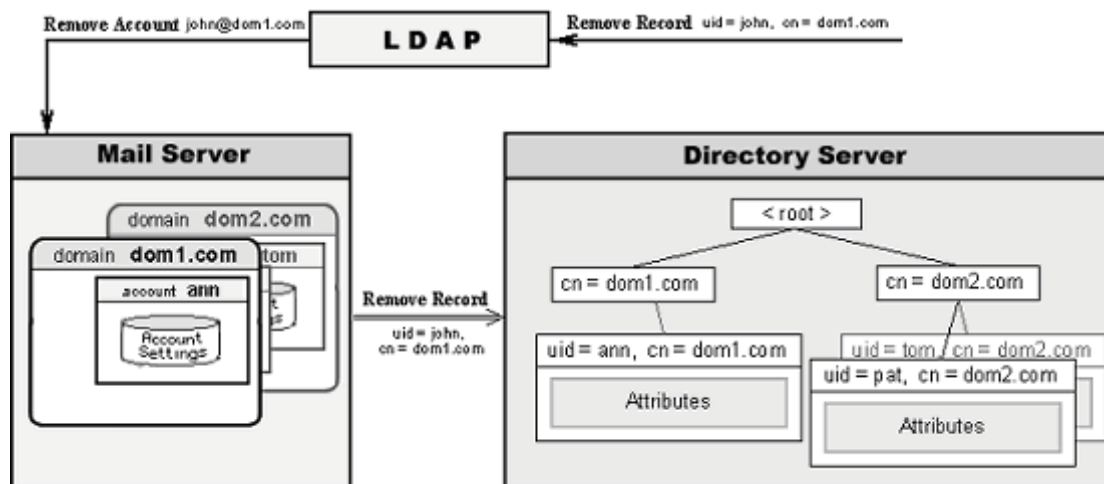
Step 5.



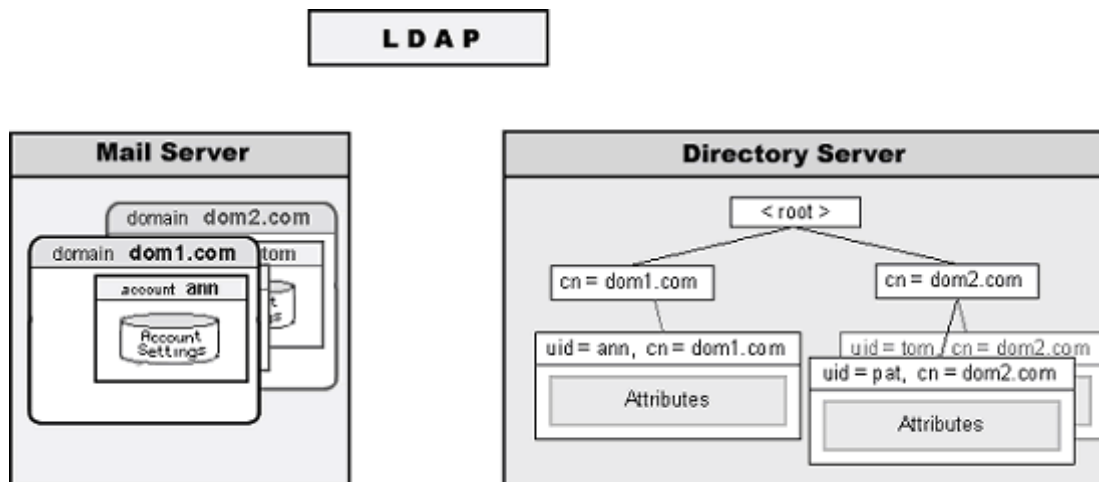
Step 6.



Step 7.



Step 8.



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.com` 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.com`, and the CommuniGate Pro Domain `dom1.com` does exist.
- Instead of performing the requested DeleteRecord operation with the Directory, the LDAP module executes the `DeleteAccount(john)` operation in the `dom1.com` Domain.
- The Account john is deleted.
- If the `dom1.com` 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.

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.

Directory-Based Domains

The CommuniGate Pro Server software implements Directory-based Domains. Directory-based Domains and 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](#).

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](#) (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 file (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.

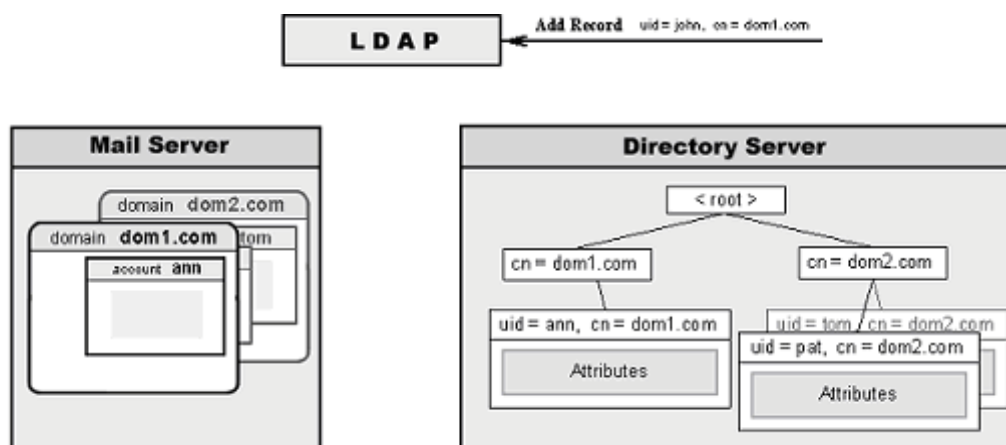
Directory records are created for aliases of Directory-based Domain Accounts.

Alias records have the same DNs as Accounts (`uid=aliasname, domain DN`).

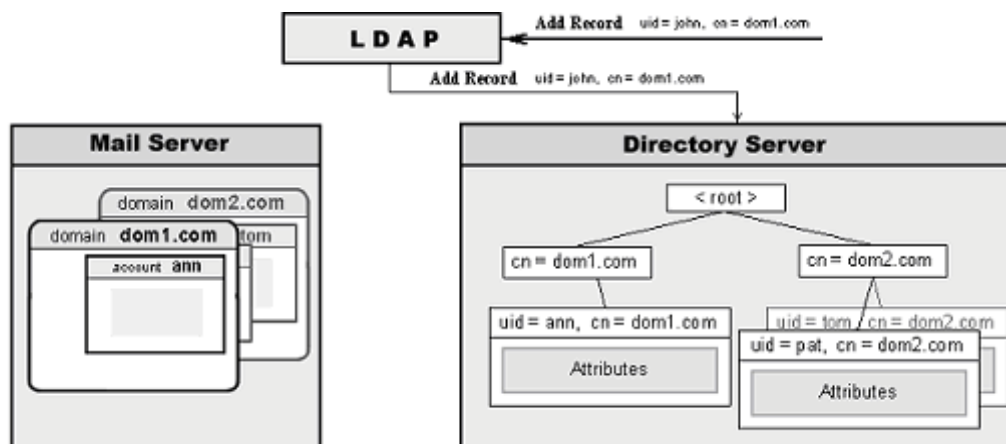
Alias records have the standard alias objectClass, and their `aliasedObjectName` attribute specifies the DN of the original account record.

The following diagram illustrates how the LDAP AddRecord operation can be used to create an Account in the Directory-based Domain:

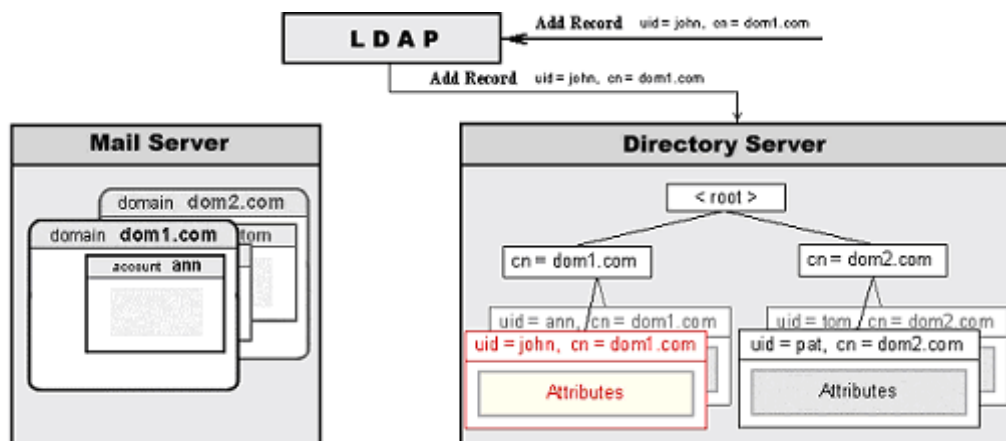
Step 1.



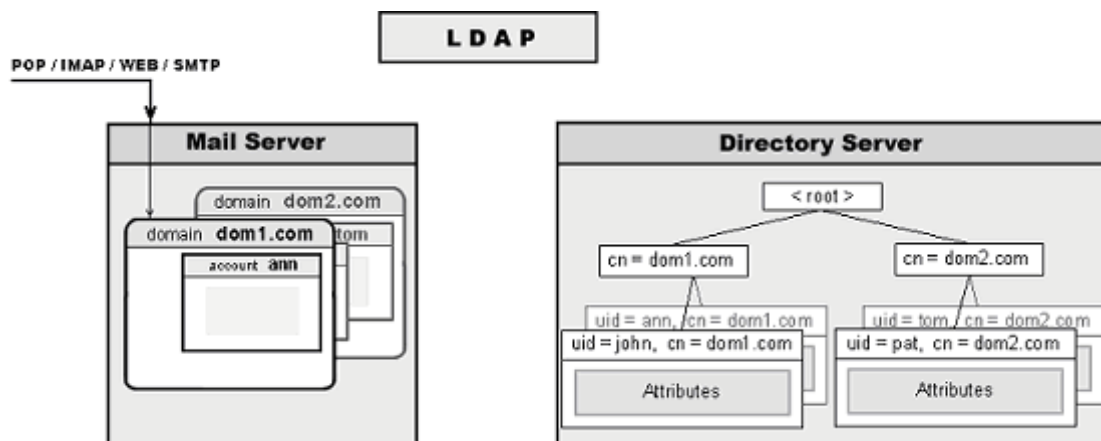
Step 2.



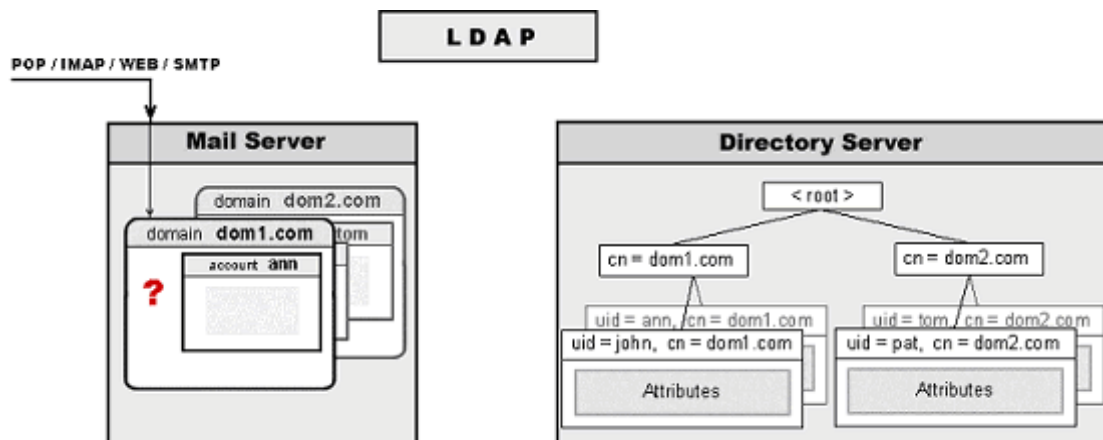
Step 3.



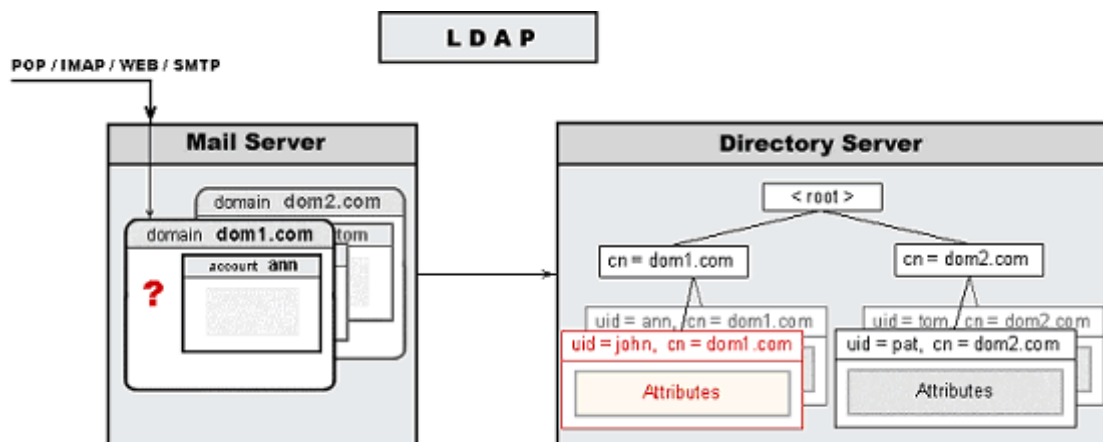
Step 4.



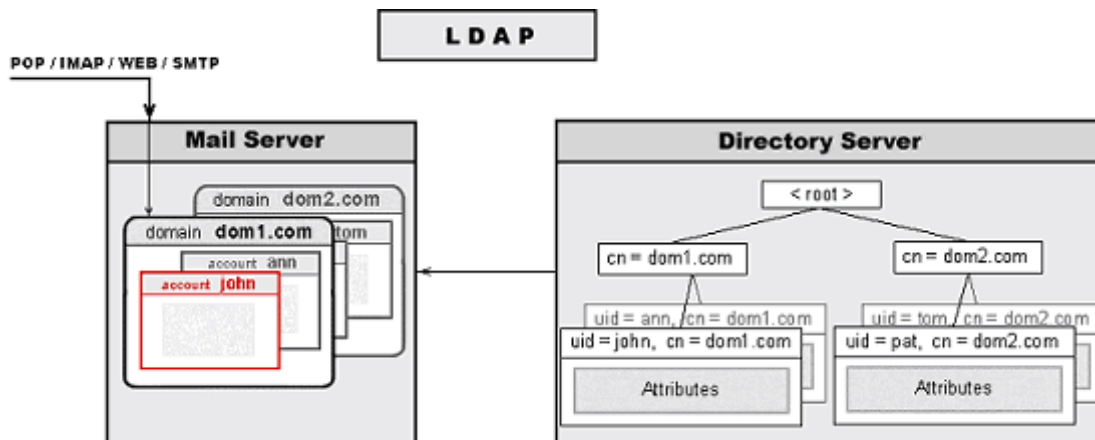
Step 5.



Step 6.



Step 7.

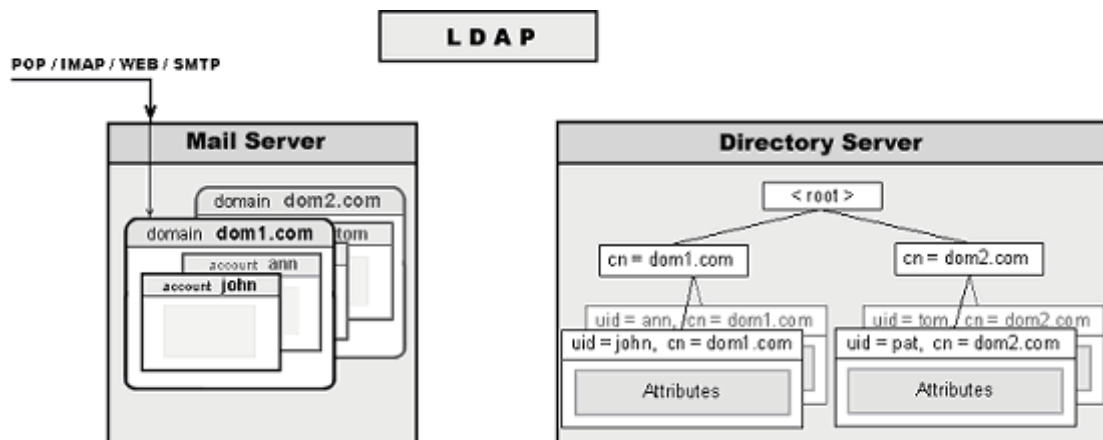


In this example:

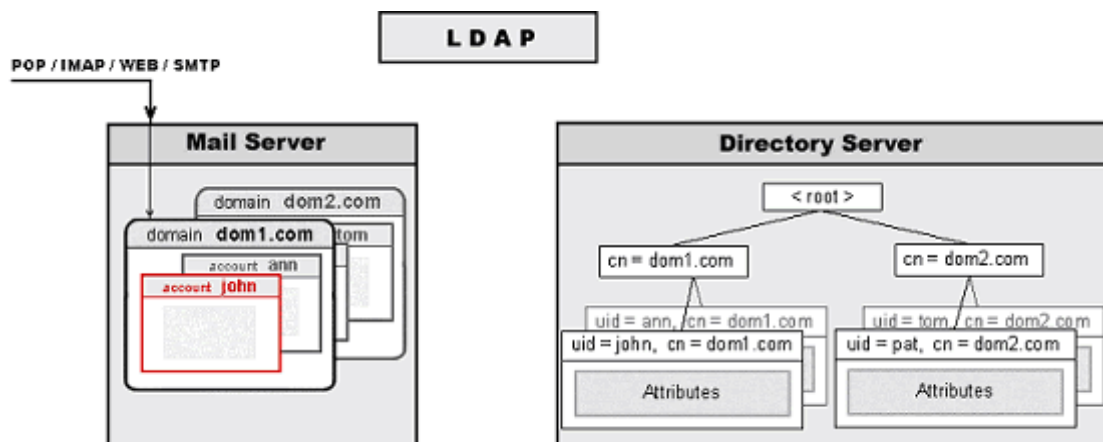
- The LDAP module received an AddRecord request from an LDAP client. The client asks the LDAP module to create a new record with the `uid=john, cn=dom1.com` DN.
- The LDAP module creates a new record in the Directory and stores all supplied attributes in it. The response is sent back to the LDAP client and the operation is completed.
- Any Mail Server component tries to open the account `john` in the Directory-Based Domain `dom1.com`. The request is sent to the Directory and the record with the `uid=john, cn=dom1.com` DN is retrieved.
- The `storageLocation` attribute in the retrieved record contains the asterisk sign. The Mail 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

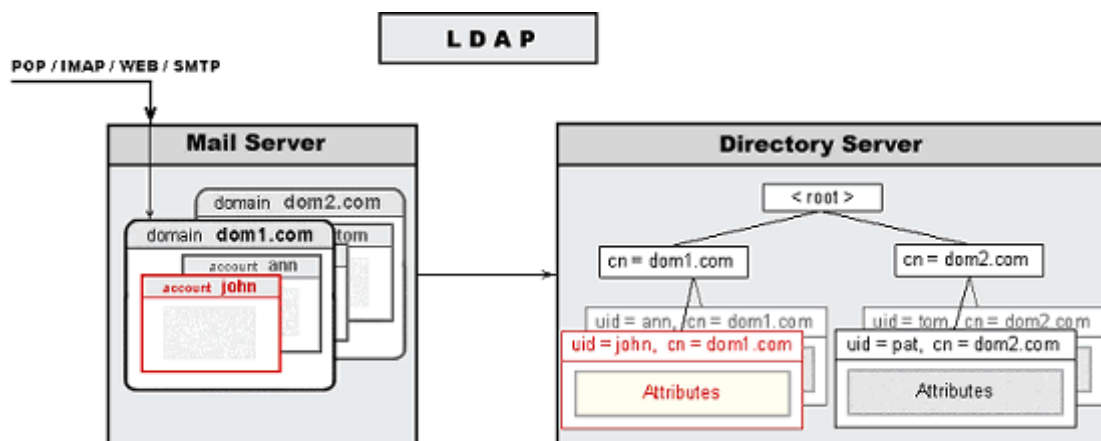
Step 1.



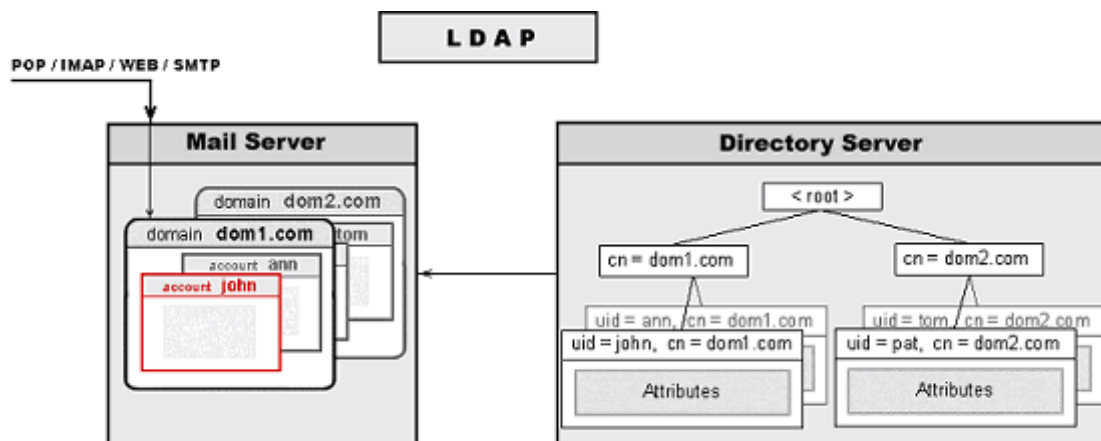
Step 2.



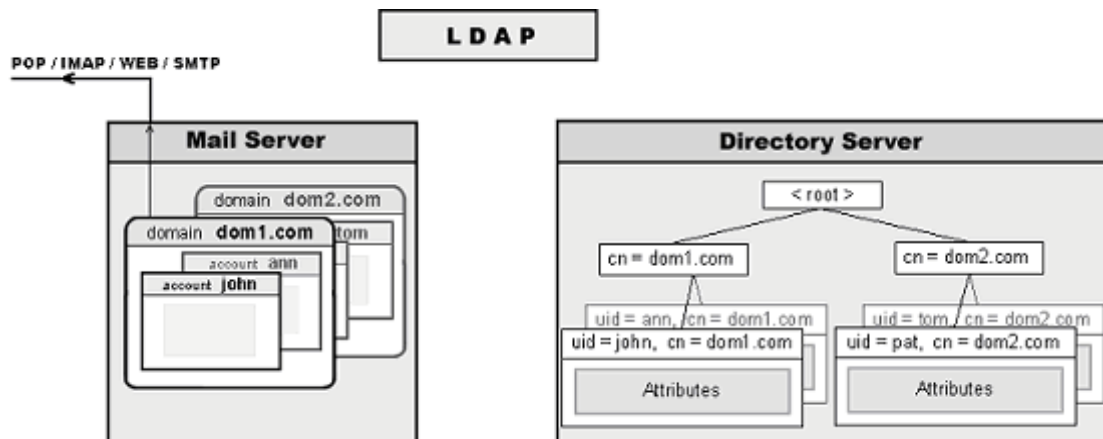
Step 3.



Step 4.

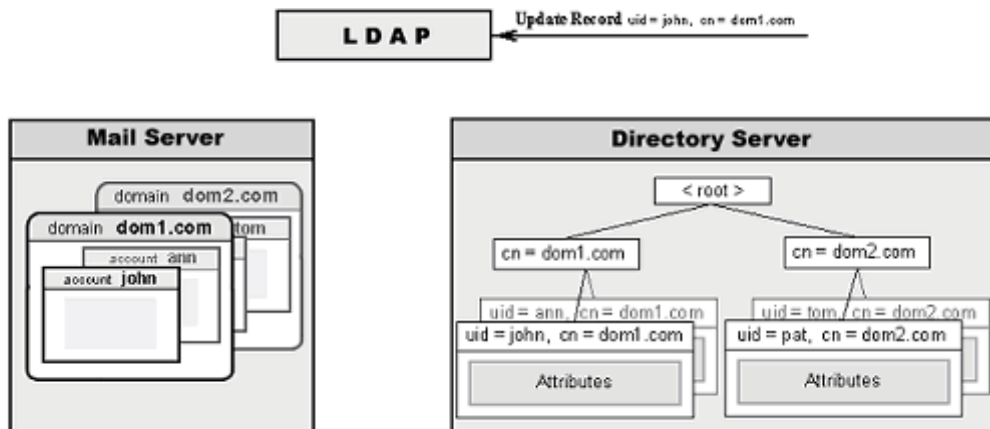


Step 5.

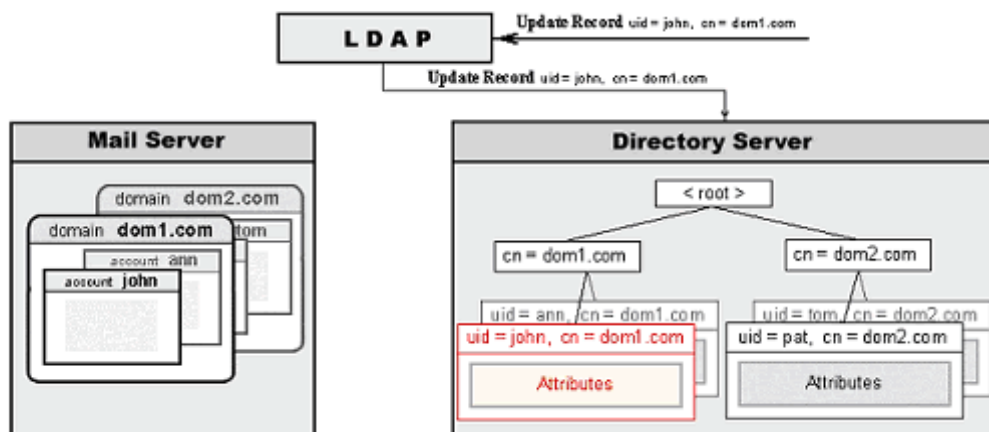


Since the Directory records are the only source of the Account settings, modifying Directory record attributes effectively modifies the Account Settings:

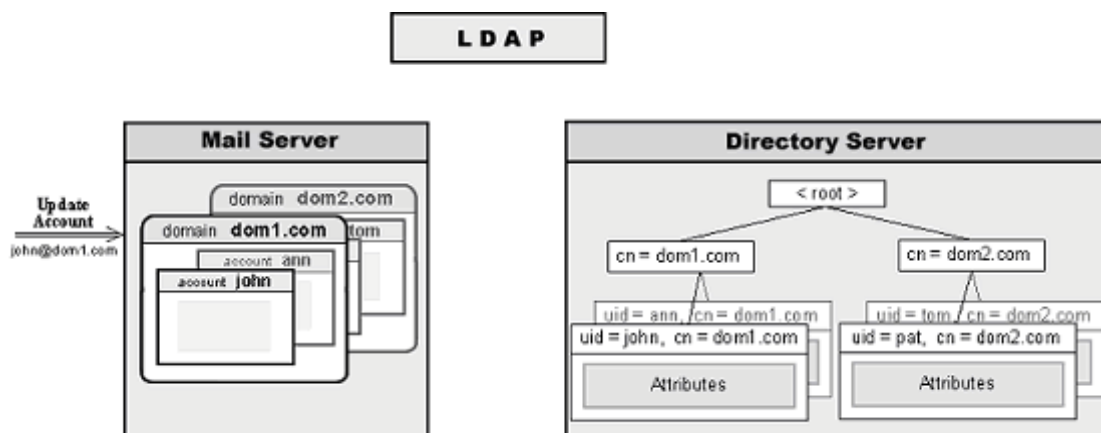
Step 1.



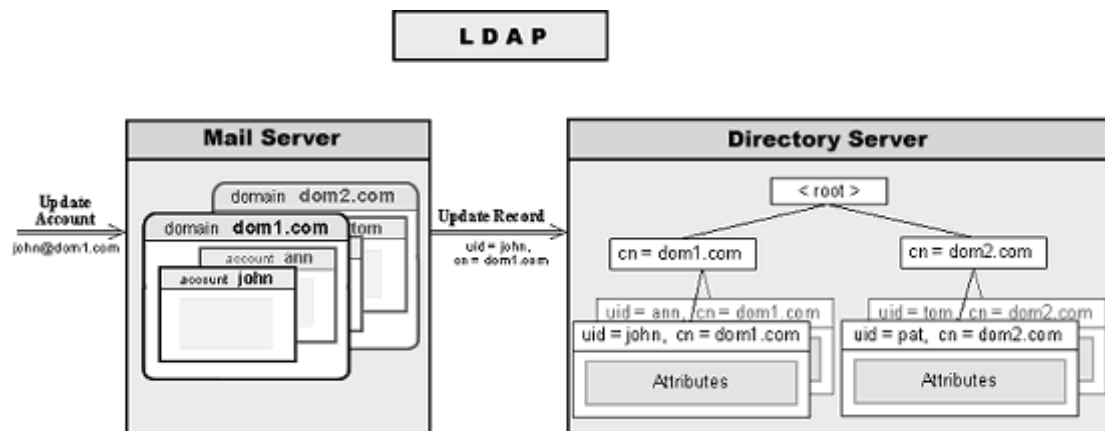
Step 2.



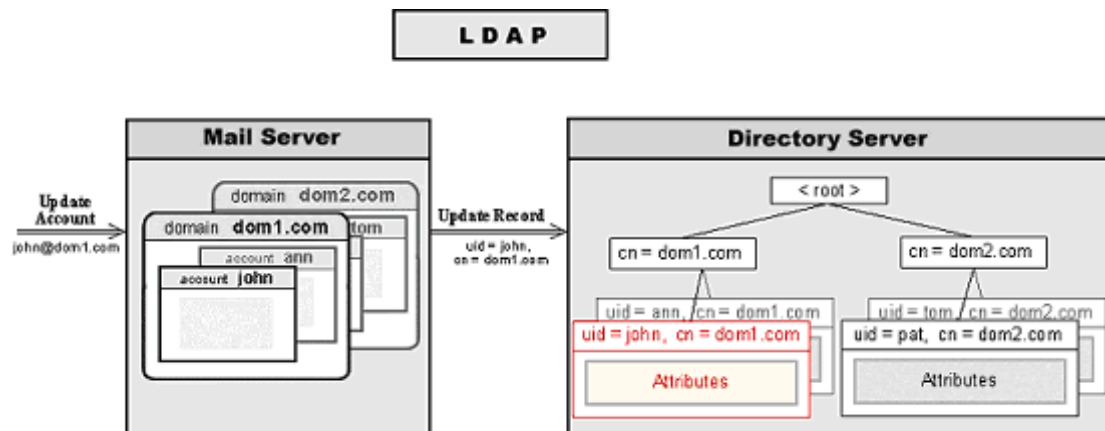
Step 3.



Step 4.



Step 5.



In this example:

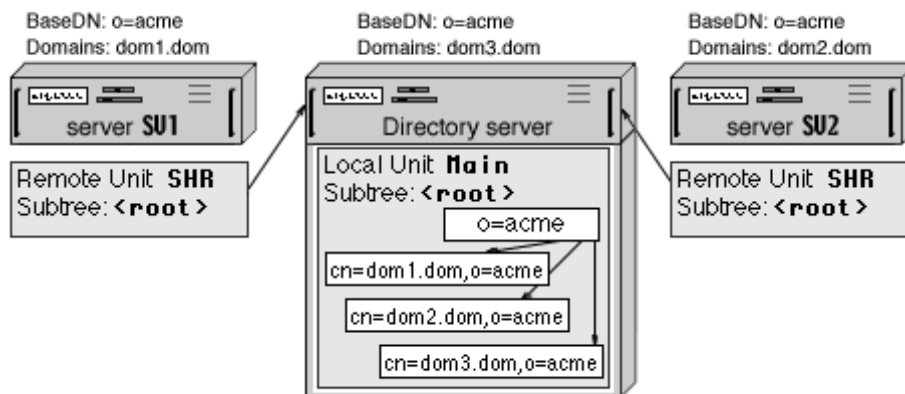
- The LDAP module received a ModifyRecord request from an LDAP client. The client asks the LDAP module to change attributes in the `uid=john,cn=dom1.com` record.
- The LDAP module tells the Directory to modify the specified record. The response is sent back to the LDAP client and the operation is completed.
- Any Mail Server component tries to open the account `john` in the Directory-Based Domain `dom1.com`. The request is sent to the Directory and the record with the `uid=john,cn=dom1.com` 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:

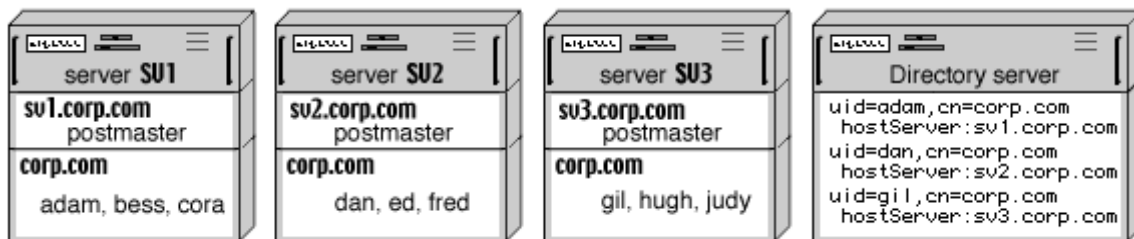


In this example:

- 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 all Domain Accounts. The Shared Domain should not be a Main Domain of any CommuniGate Pro Server:



In this example:

- Three CommuniGate Pro Servers (with the `sv1.corp.com`, `sv2.corp.com`, and `sv3.corp.com` Main Domains) all have the same `corp.com` Secondary Domain. Some Accounts are created in the `corp.com` 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.com` 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.com` 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.com` Servers.

When any Server creates an Account and places a record into the Shared Directory, it stores the Server Mail Domain name as the record `hostServer` attribute.

The Shared Directory can be used to route Shared Domain mail to the proper location (Server). After you enable the `Directory-Based Routing` Setting in the CommuniGate Pro General->Cluster Settings, 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 remote 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.

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. In the [Distributed Domain Settings](#), set the Mail To Unknown option to

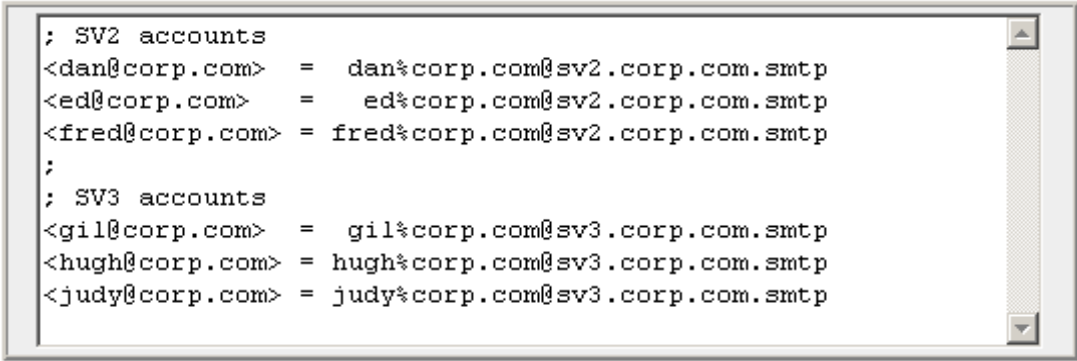
```
Reroute To: *%domain.com@mainserver.smtp
```


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 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.com> = dan%corp.com@sv2.corp.com.smtp
<ed@corp.com> = ed%corp.com@sv2.corp.com.smtp
<fred@corp.com> = fred%corp.com@sv2.corp.com.smtp
;
; SV3 accounts
<gil@corp.com> = gil%corp.com@sv3.corp.com.smtp
<hugh@corp.com> = hugh%corp.com@sv3.corp.com.smtp
<judy@corp.com> = judy%corp.com@sv3.corp.com.smtp
```

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.com, ed-uk@corp.com, fred-uk@corp.com, etc.), routing for all SV2 Accounts can be specified with one Router record:

<*-uk@corp.com> = *-uk%corp.com@sv2.corp.com.smtp



Clusters

When your site serves more than 150,000-200,000 accounts, or when you expect a really heavy IMAP/WebMail/MAPI/SIP traffic, you should consider using a multi-server *Cluster configuration*.

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 in each, 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 signalling
- POP3 access
- IMAP access
- WebUser Interface access
- XIMSS access
- FTP
- RADIUS
- TFTP

- HTTP access to File Storage (including uploading)
- ACAP access
- PWD access and remote administration

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
- handle most of the SMTP relaying operations themselves
- 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

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 Web-Mail 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 Server, and enter all Frontend and Backend Server IP addresses. Backend 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 Members			
This Server Cluster Address:	On restart	Active	
	[192.168.0.5]	[192.168.0.5]	
Backend Server Addresses:	192.168.0.4-192.168.0.8; this is a comment		
	<div> <div>◀</div> <div>▶</div> <div>⌵</div> <div>⌶</div> </div>		
Frontend Server Addresses:	192.168.0.1-192.168.0.3		
	<div> <div>◀</div> <div>▶</div> <div>⌵</div> <div>⌶</div> </div>		
Dynamic Cluster Locker Log:	Problems	Admin Connections:	Enabled

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 the `first IP` value is selected, the Server selects the first address from the list of Server Local IP Addresses.

If you change this setting value, the new value will be in effect only after Server restart.

Admin Connections

See the [Security Issues](#) section below.

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 mail services, change the Backend Server Ports val-

ues.

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.

Backend Port		Cache	Backend Port		Cache
PWD:	<input type="text" value="106"/>		Delivery:	<input type="text" value="25"/>	<input type="text" value="15"/> ▼
POP:	<input type="text" value="110"/>		IMAP:	<input type="text" value="143"/>	
HTTPU:	<input type="text" value="8100"/>		HTTPA:	<input type="text" value="8010"/>	
			ACAP:	<input type="text" value="674"/>	
Log Level		Cache	Log Level		Cache
Object Admin:	<input type="text" value="Problems"/> ▼	<input type="text" value="5"/> ▼	Mailboxes:	<input type="text" value="Low Level"/> ▼	<input type="text" value="15"/> ▼
Cluster Admin:	<input type="text" value="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.

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 mem-

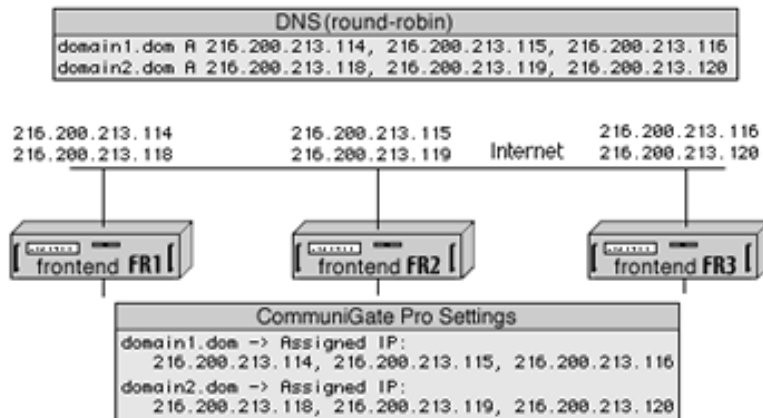
bers. The Cluster Mailbox manager has its own Log Level setting.

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:

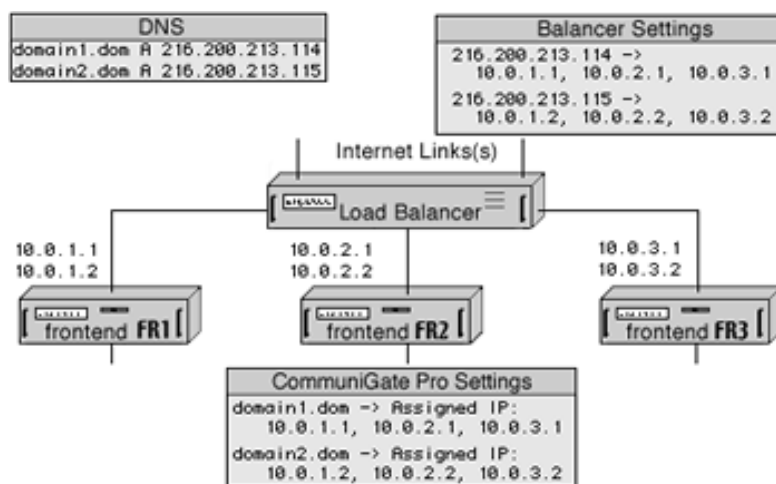


In this example, the Cluster is serving two Shared Domains: `domain1.dom` and `domain2.dom`, and the Clus-

ter 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.

Security Issues

The Frontend-Backend topology allows you to protect the site information and Backend Servers not only when a Frontend Server crashes because of some type of network attack, but even if the Frontend Server OS is "cracked" and an intruder gets the complete ("root") access to the Frontend Server OS using a security hole in that OS.

To protect the site from these "cracks":

- Do not use the Frontend Servers to administer the Shared Domains as a Frontend Server administrator. In this case you can disable the Admin Connections option on the Cluster page of all Backend Servers.
- Enable the Admin Connections option on the current Cluster Controller and Backup Controller Servers only when you are adding a new Server to the Cluster. When that new Server is up and running, you can disable the Admin Connections option on the Cluster Controllers.

These measures do not cause any problem for your users that have the domain administrator rights and want to administer their Shared Domains (using WebAdmin Interface or CLI). They also do not cause any problem for your regular users that want to use the PWD module to update their passwords.

Cluster Configuration Details

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 sign (*). 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 (,) sign.

RPOP

RPOP activity takes place on Backend servers in a Static Cluster, and on the Cluster Controller in a Dynamic Cluster. 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.

FTP

The FTP module does not "proxy" connections to Backend servers. Instead, it uses CLI to manage Account Personal File Sites 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.

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.

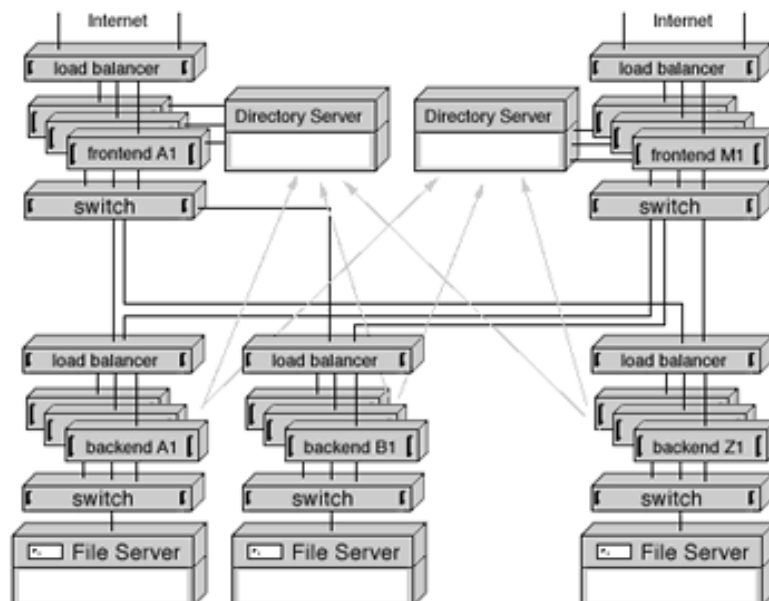
Postmaster Account

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 "Can Modify All Domains and Accounts Settings" [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 Serv-

ers 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.



Static Clusters

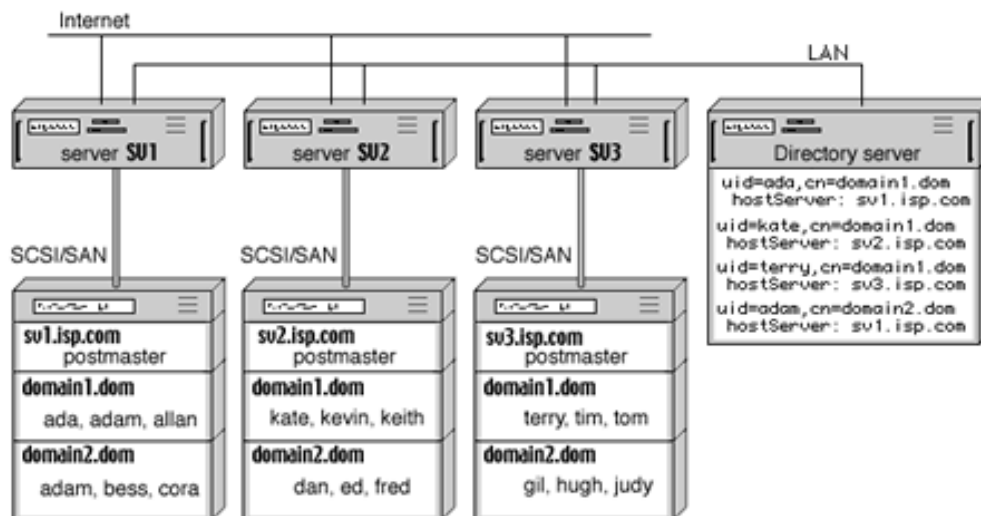
Static Clusters can be used to handle extremely large (practically unlimited) sites, providing 24x7 site access.

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:

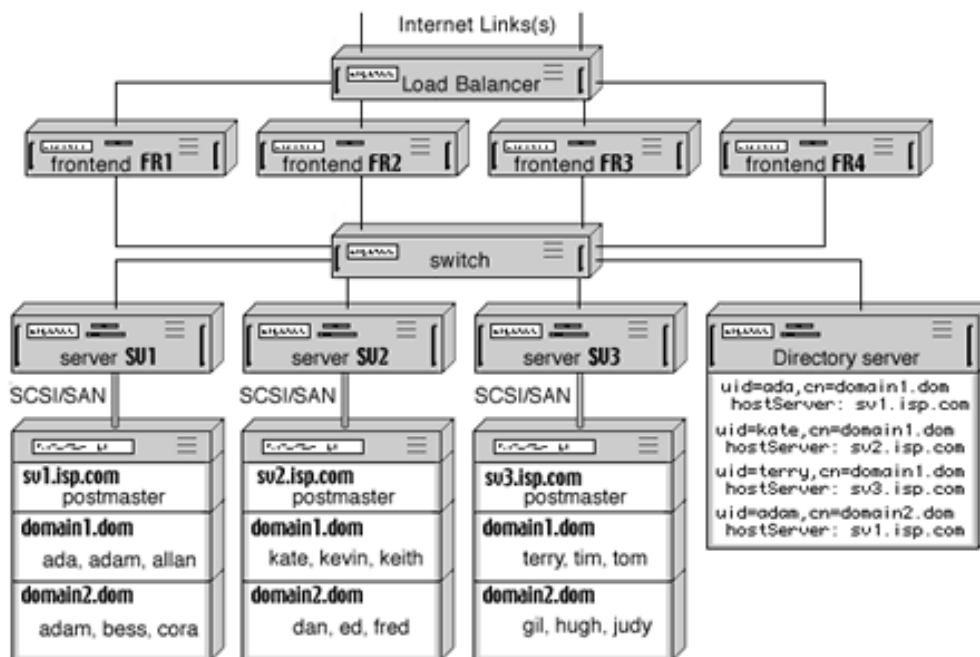
Static Clustering	
Static Member Name	Member Address
sv1.isp.com	216.200.213.115
sv2.isp.com	216.200.213.116
sv3.isp.com	216.200.213.117

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.



Dynamic Clusters

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 Messaging servers can employ file servers for account 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: *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 10-Server cluster is even more difficult than administering 10 independent Servers.

The CommuniGate Pro software supports the [External 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.

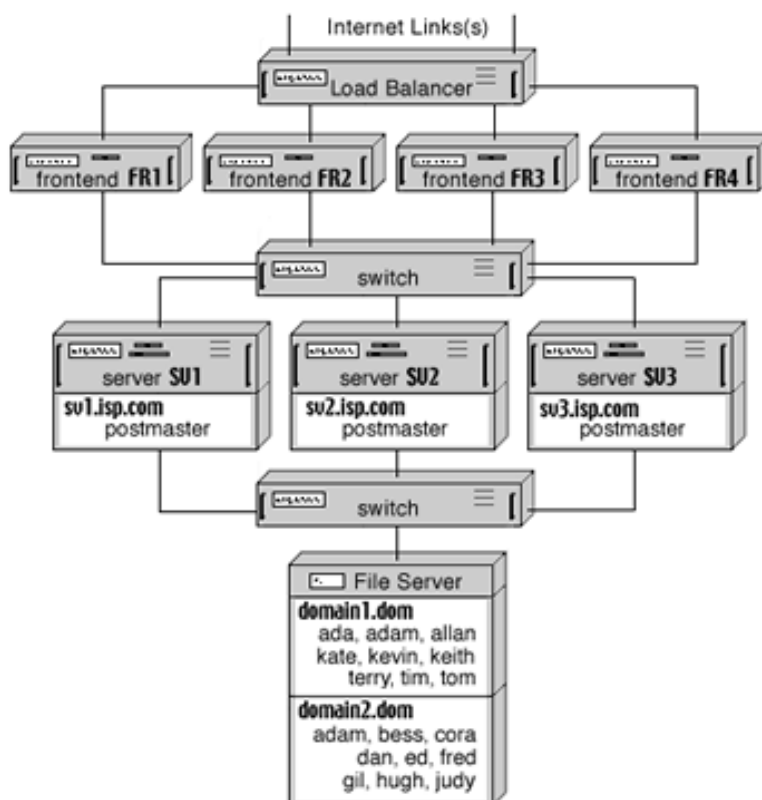
The Cluster Controller collects the load level information from the Backend Servers. When a Frontend Server receives a session request for an Account not currently opened on any Backend Server, the Controller directs the Frontend Server to the least loaded Backend Server. This second-level load balancing 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 extremely useful for the CommuniGate Pro Dynamic Cluster implementations:

- 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.

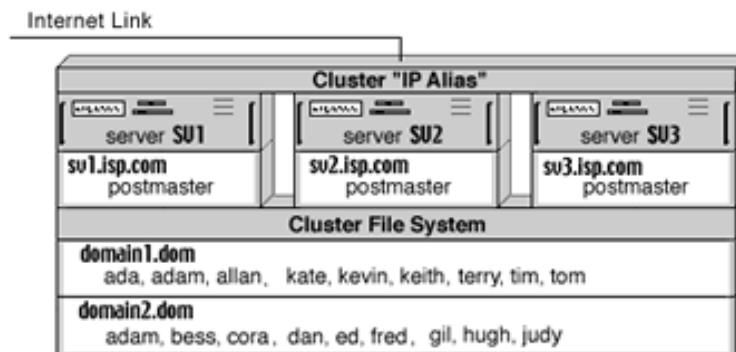
The SAN protocols are very effective for file transfers, and Cluster File Systems can provide better performance than Network File Systems.

The Cluster File Systems can also provide better reliability than single-server NFS solutions (where the NFS server is a single point of failure).

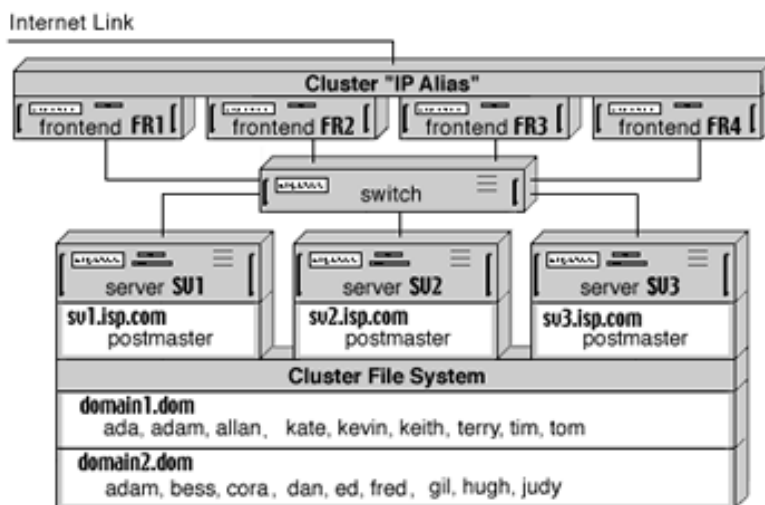
See the [Storage](#) section for more details.

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:



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

$$2 * (\text{number of Backend servers} + \text{number of Frontend servers})$$
 Since additional PWD connections can be opened by Frontend and Backend servers to serve administra-

tor and user requests, it is better to increase the number of channels by:

$5 * (\text{number of Backend servers}) + 3 * (\text{number of Frontend servers})$

- 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 SharedDomain 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 domains 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 CommuniGatePro 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 CommuniGatePro 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 [SMTP queue release](#) (via ETRN or Wakeup E-mail).
- Cluster-wide Authentication token (*nonce*) synchronization.

Withdrawing Servers from a Dynamic Cluster

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 10-20 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

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".

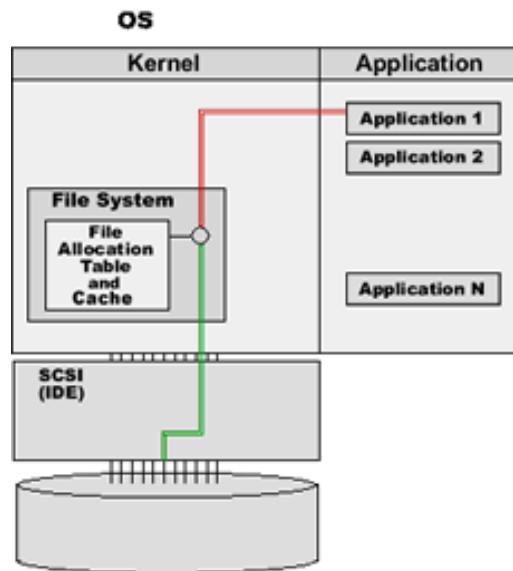
There are many different File Systems, and they use various methods and algorithms, but the same basic functions are present in most File Systems:

- 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."

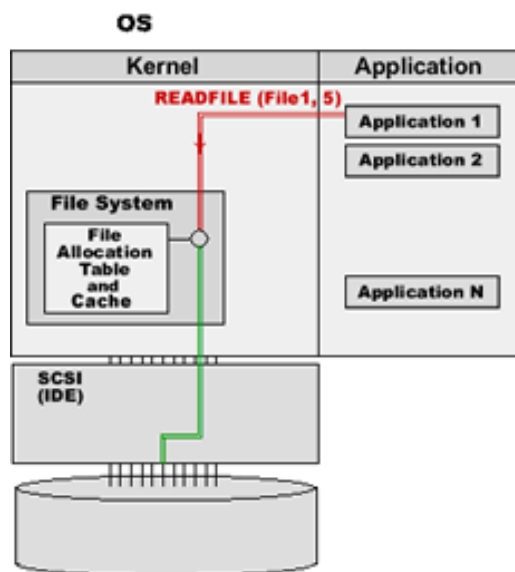
When file information is read from storage, it passes it to the application program and also copies it into the "cache buffers." When the same (or some other) application needs to read the same portion of the cached file, the File System simply retrieves that information from its cache buffers instead of re-reading it from the Storage System.

The following figure illustrates how a File System works:

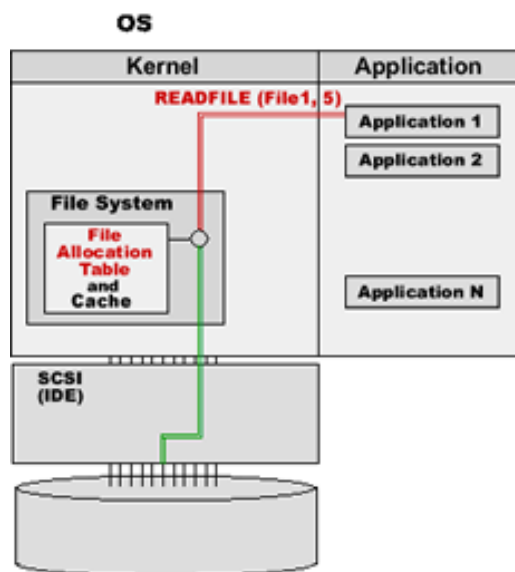
Step 1.



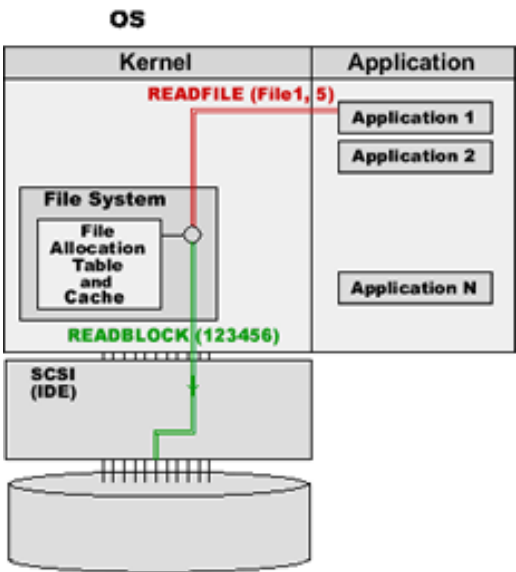
Step 2.



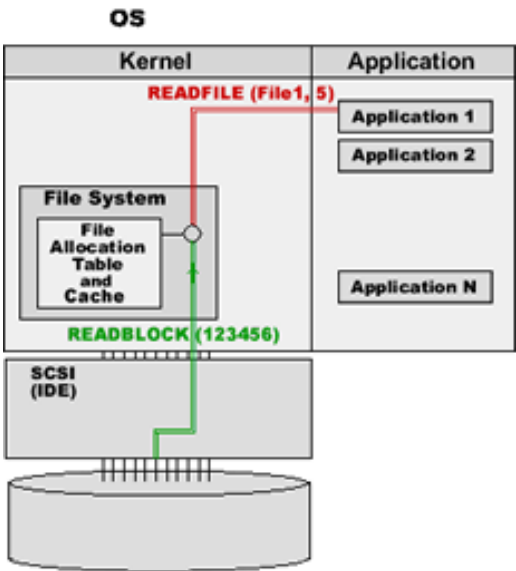
Step 3.



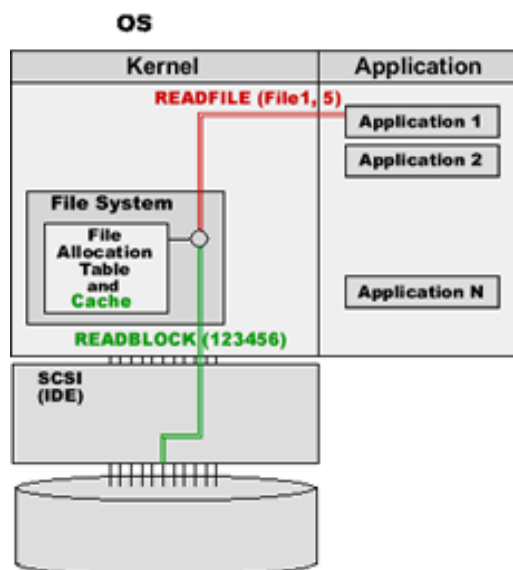
Step 4.



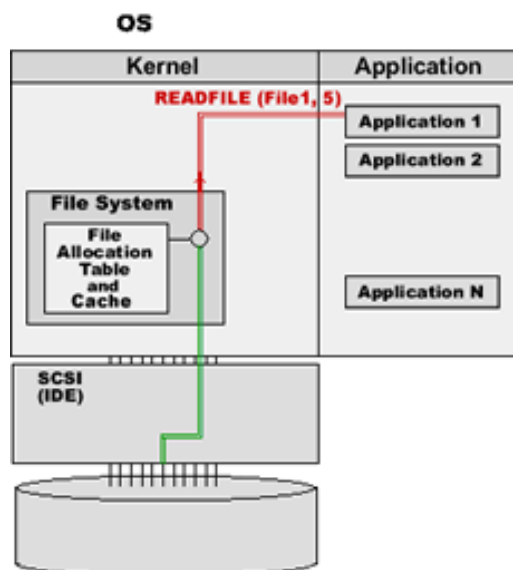
Step 5.



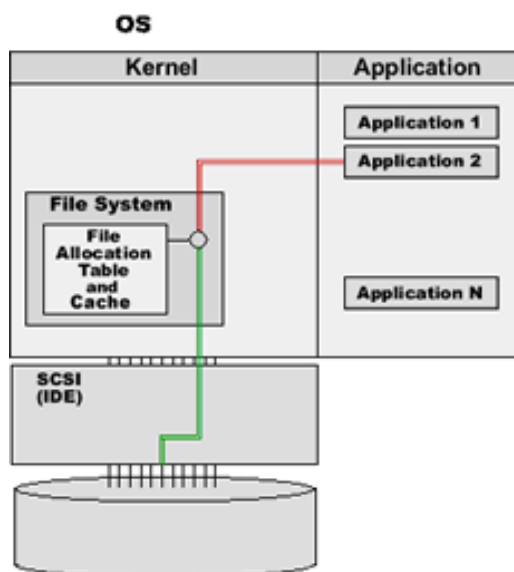
Step 6.



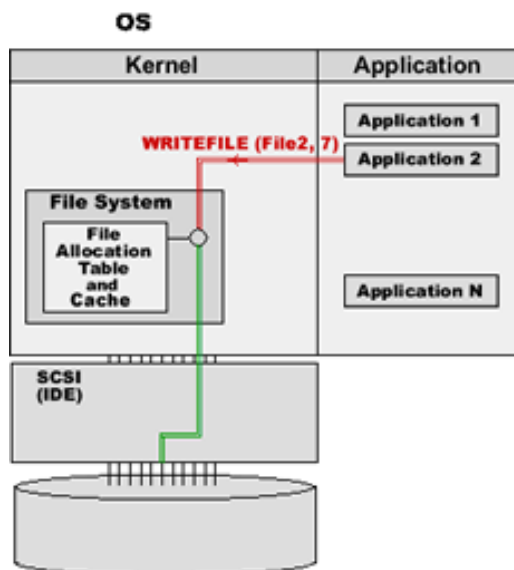
Step 7.



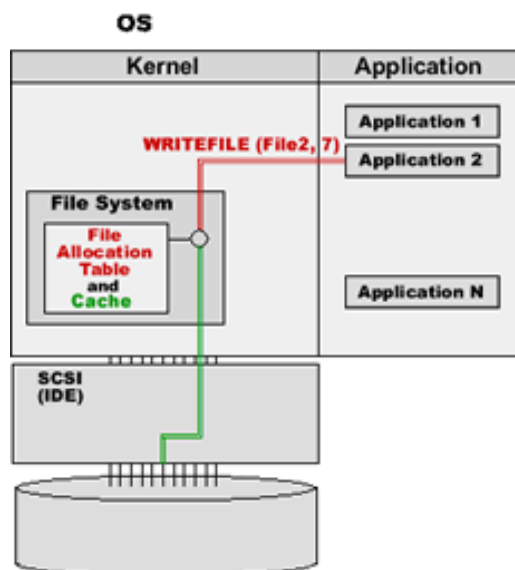
Step 8.



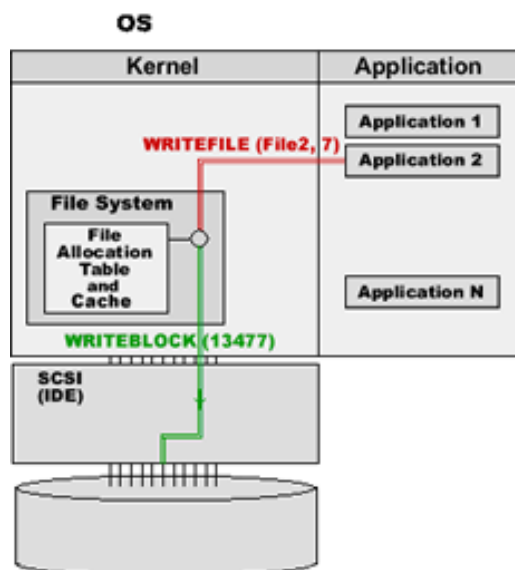
Step 9.



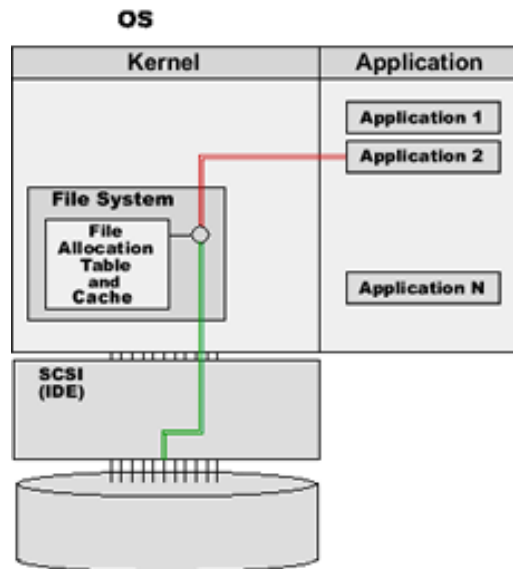
Step 10.



Step 11.



Step 12.



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 READ-BLOCK(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 WRITE-BLOCK(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.

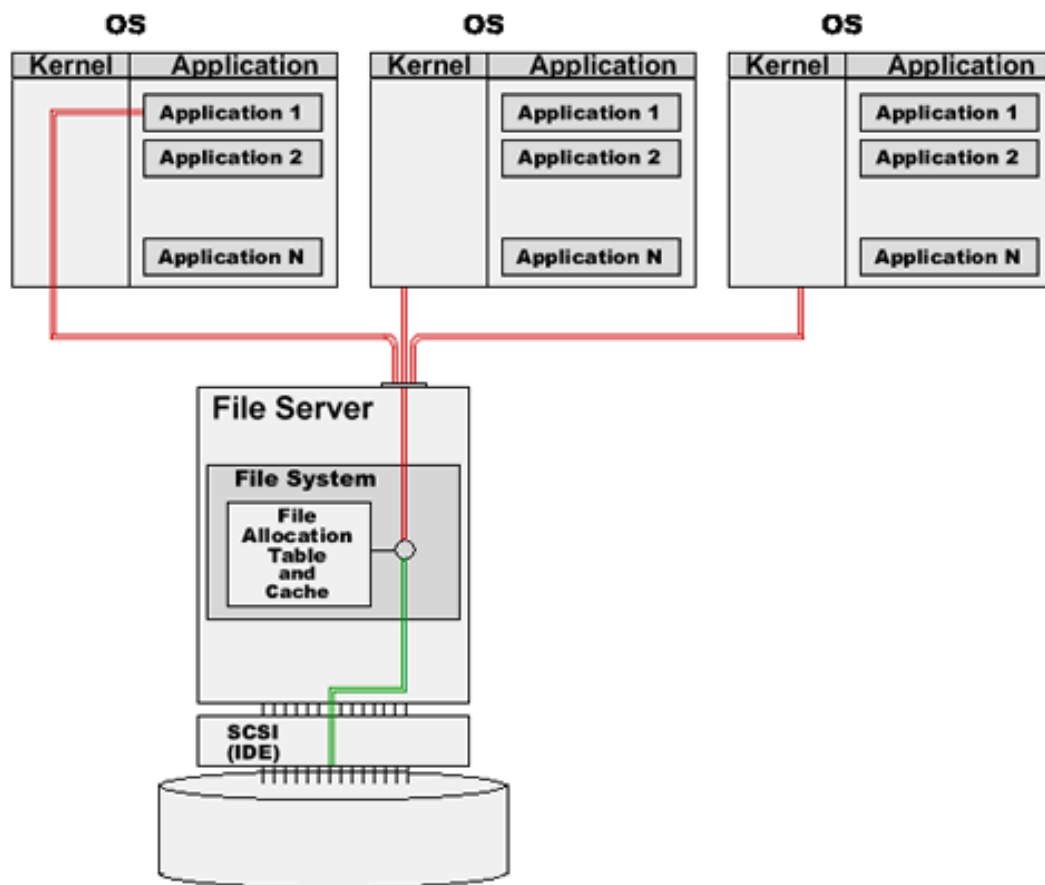
Network File System (NAS)

When server computers need to use the same data, a Network File System (also called NAS, or Network Attached Storage) can be used.

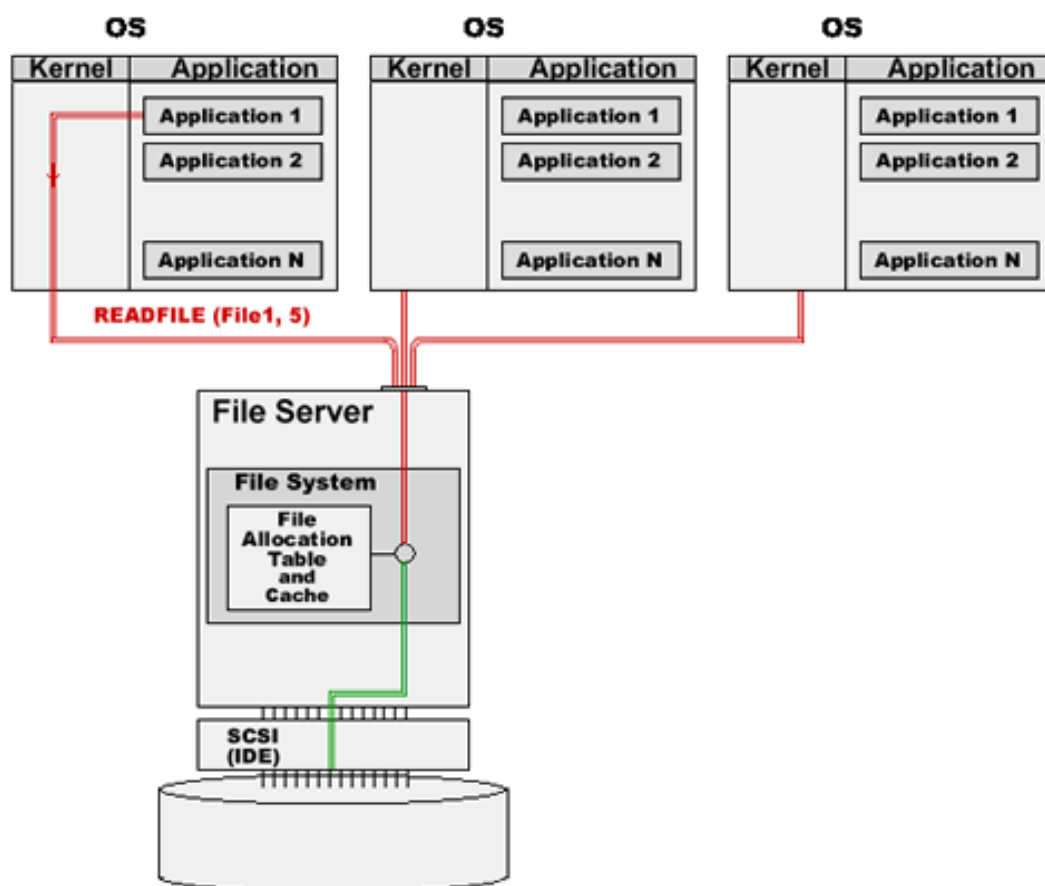
The Network File System is implemented using a *File Server* and a *network*. The File Server is a regular computer or specialized OS that has a regular File System and regular disk devices controlled with this File System.

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:

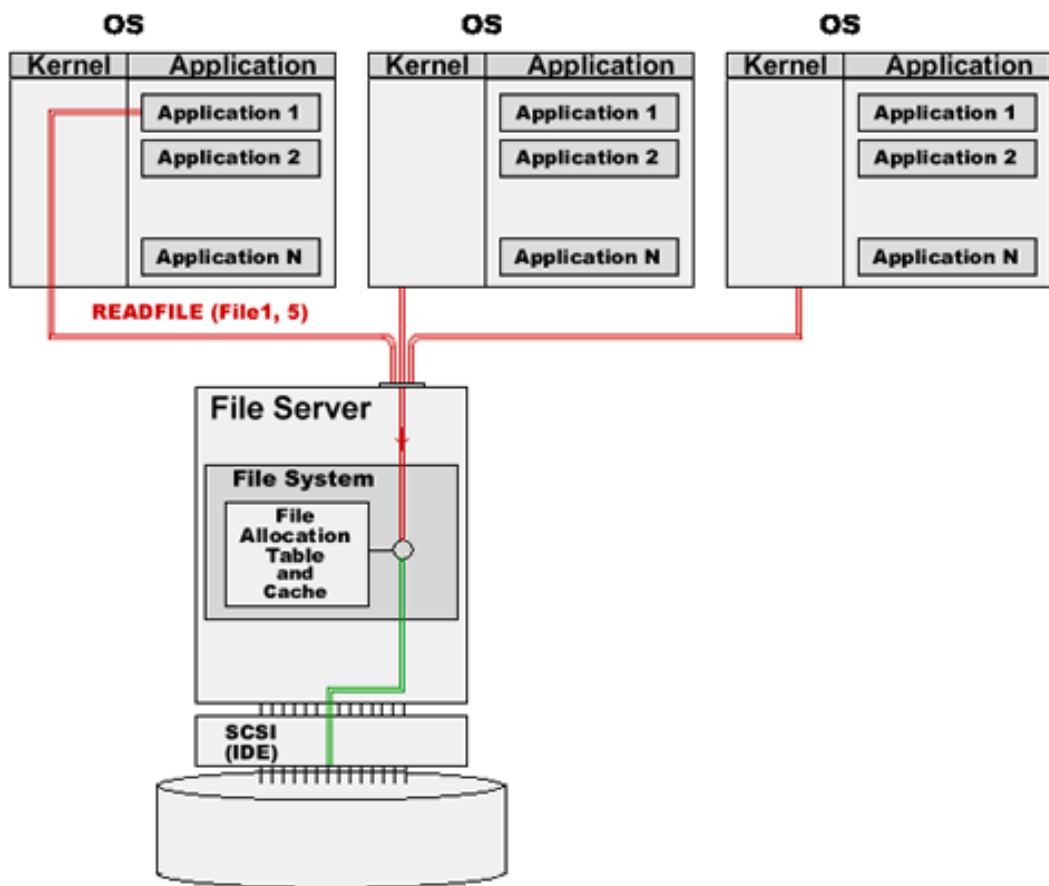
Step 1.



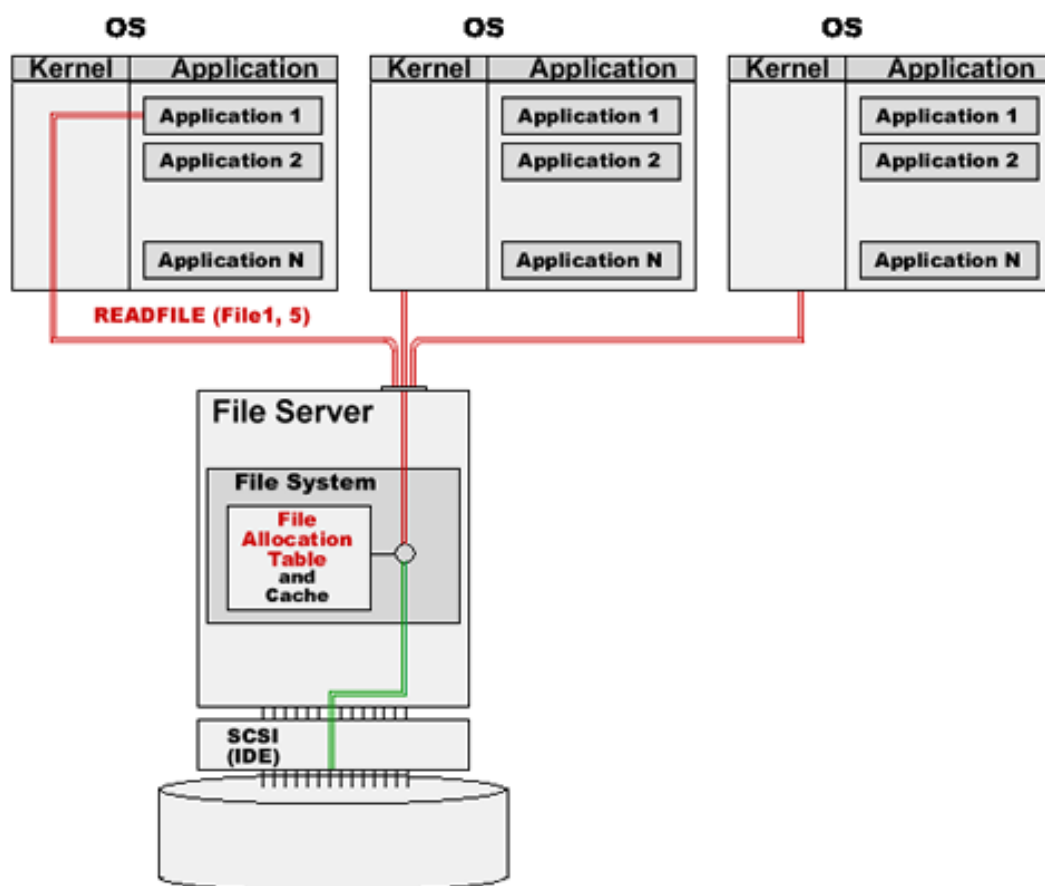
Step 2.



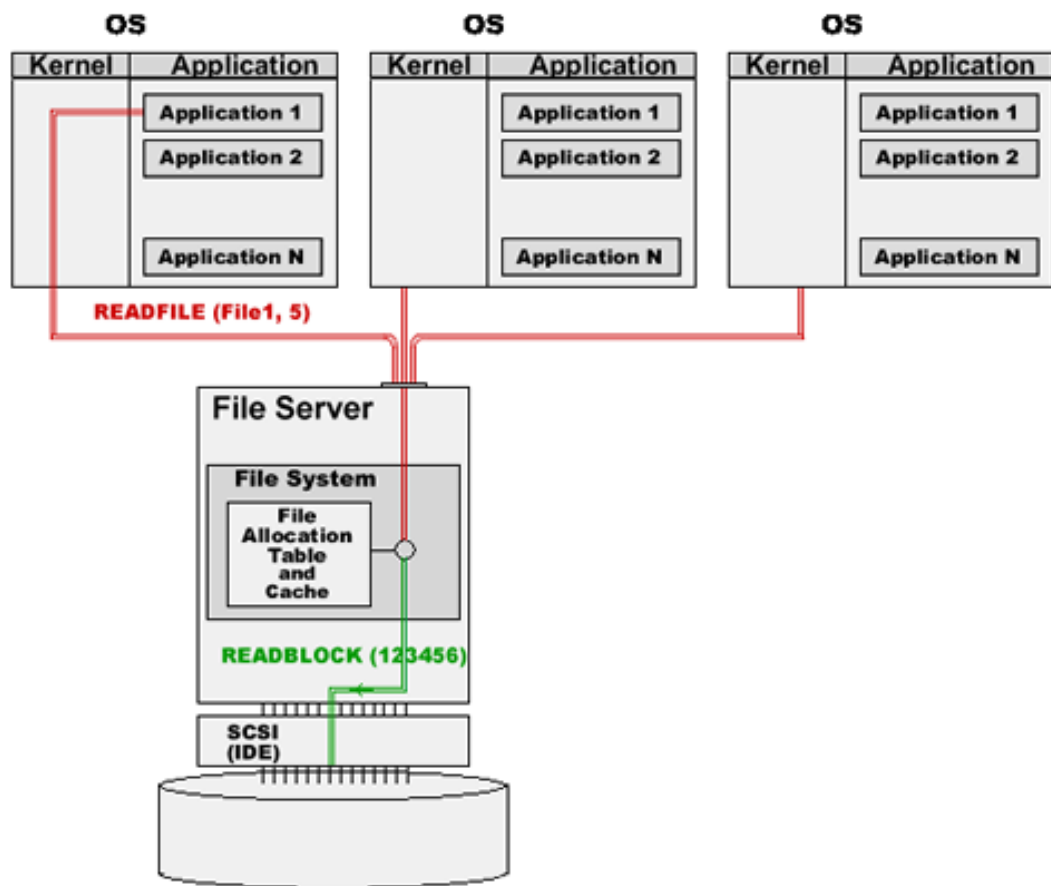
Step 3.



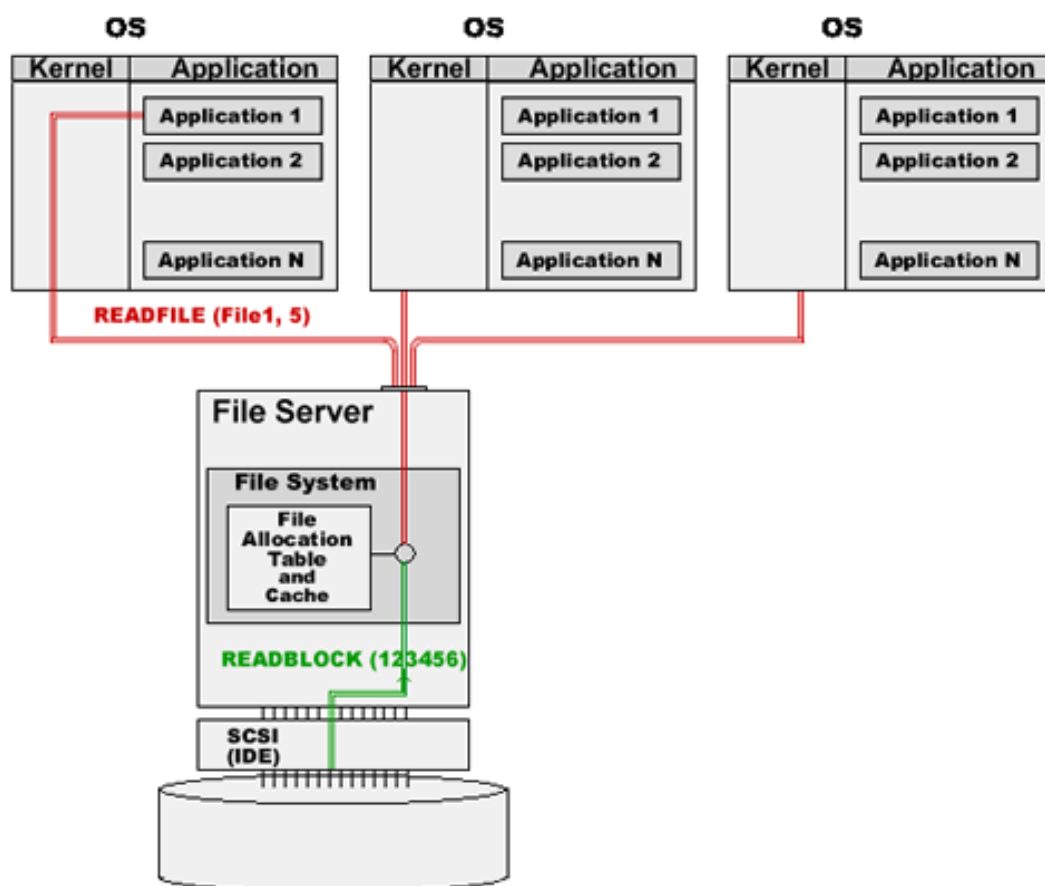
Step 4.



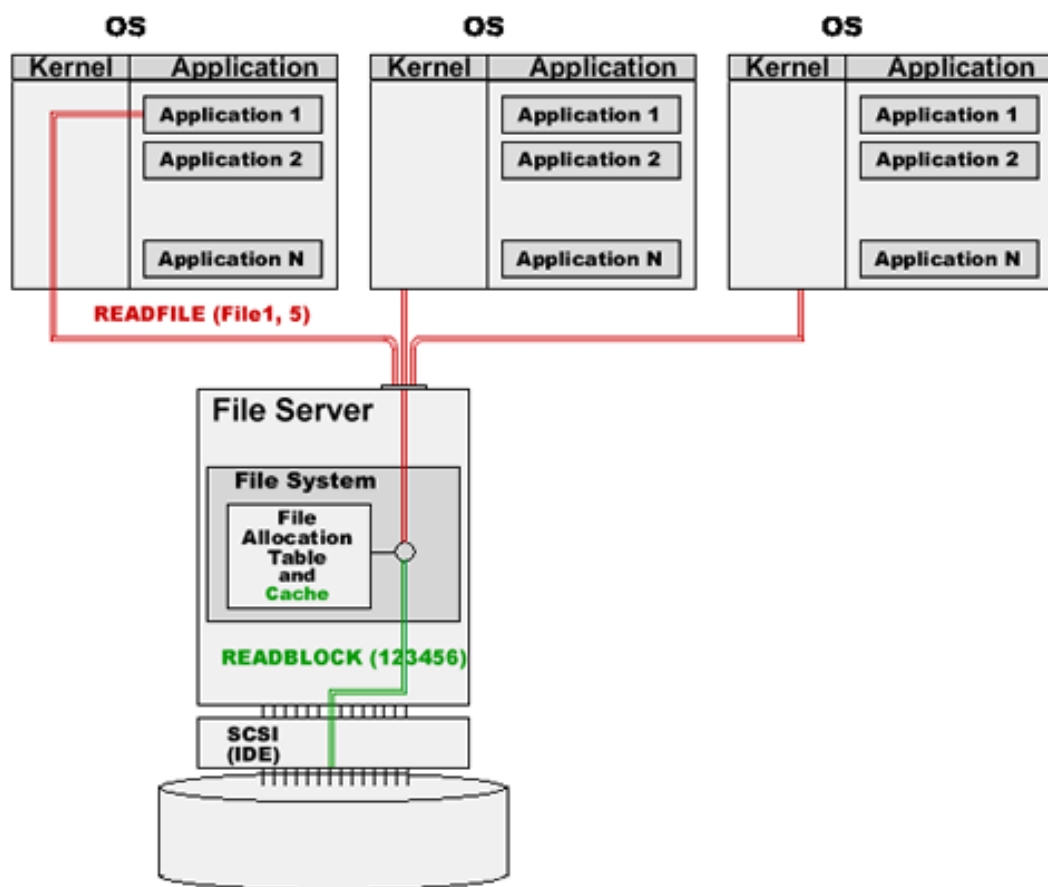
Step 5.



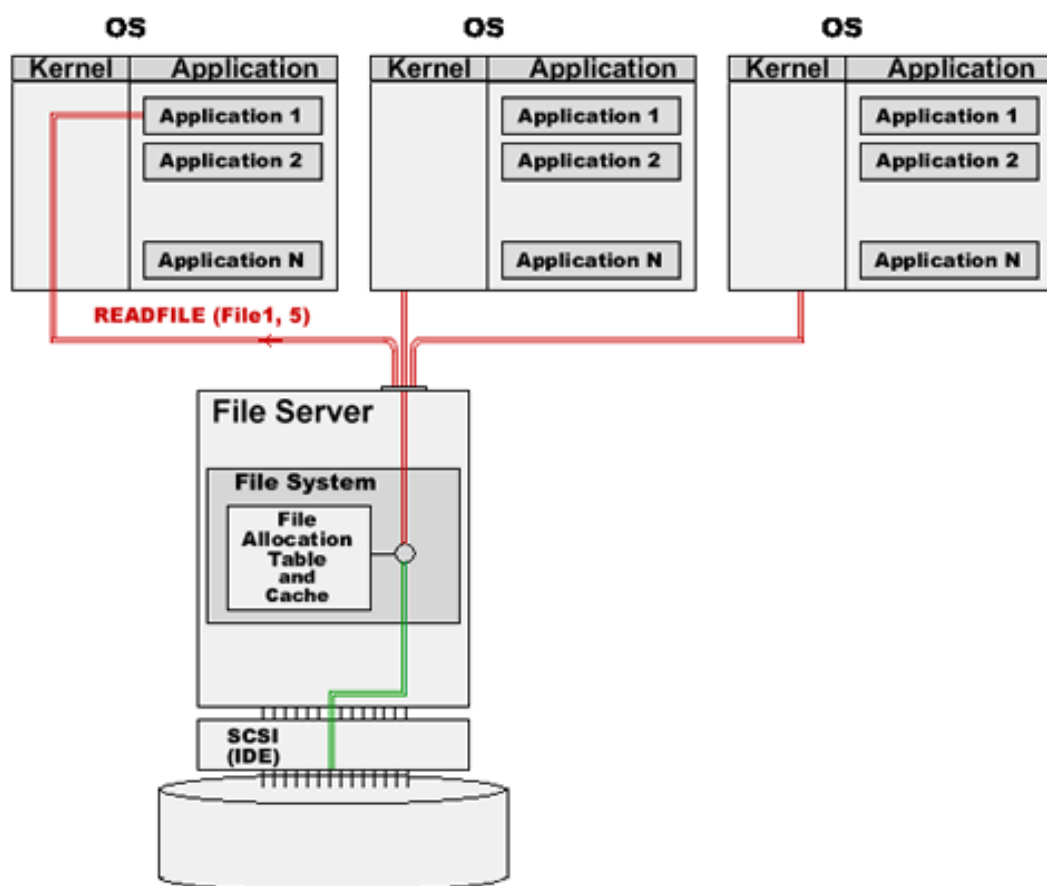
Step 6.



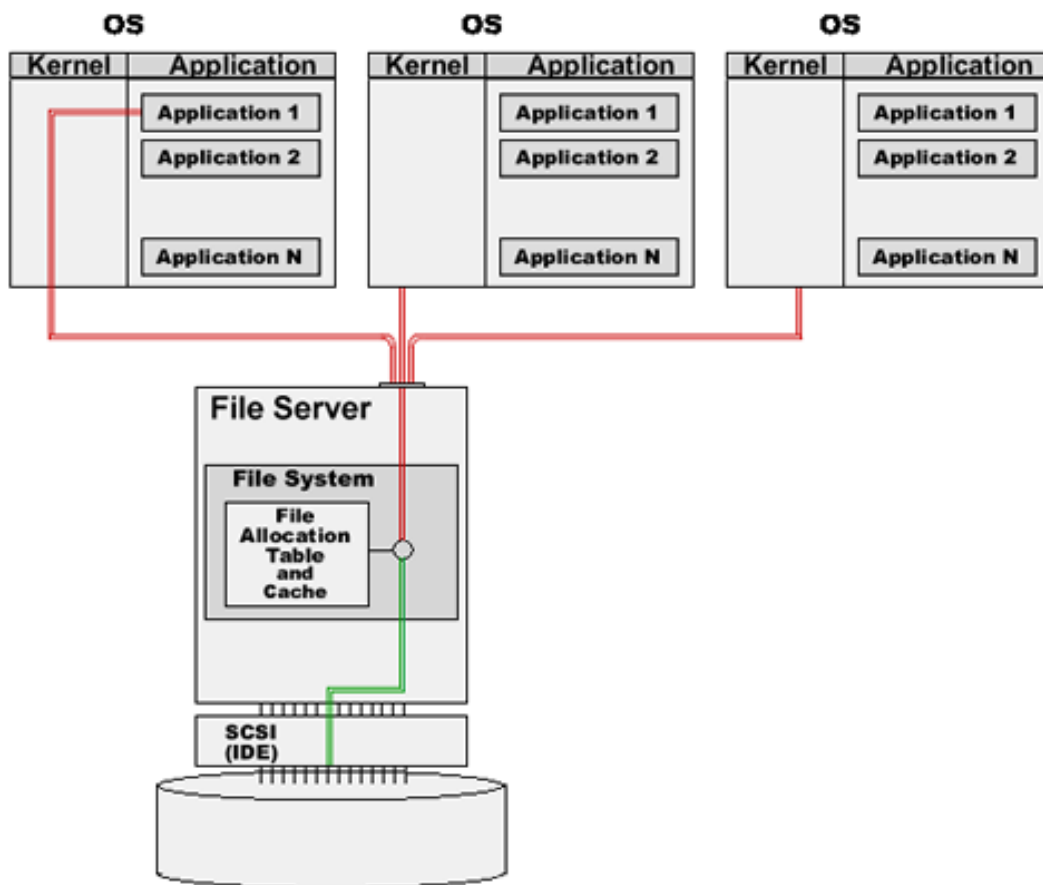
Step 7.



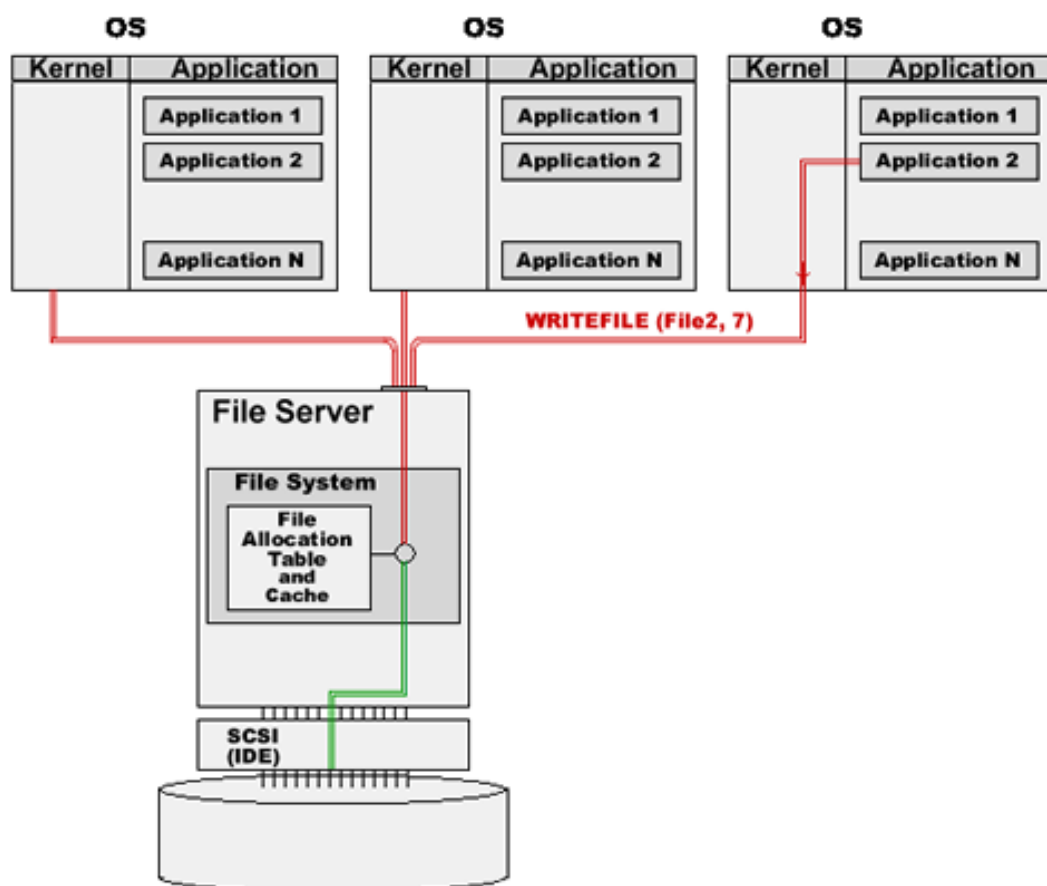
Step 8.



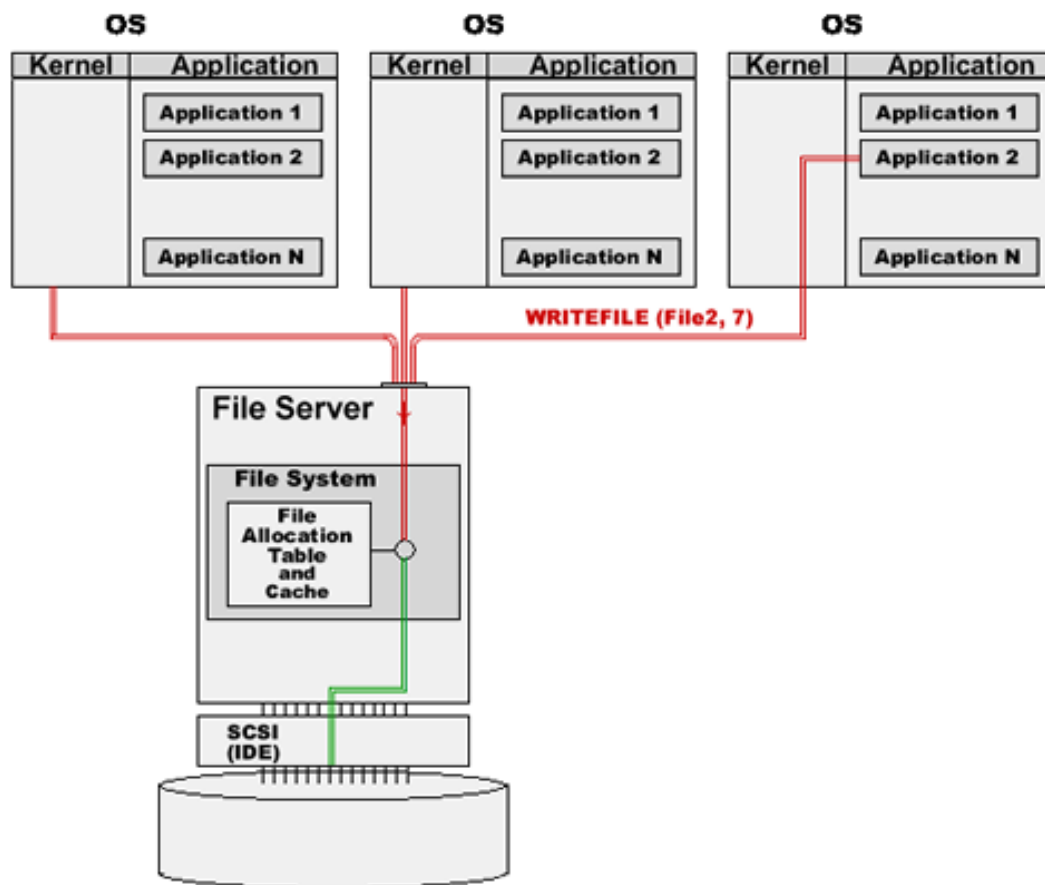
Step 9.



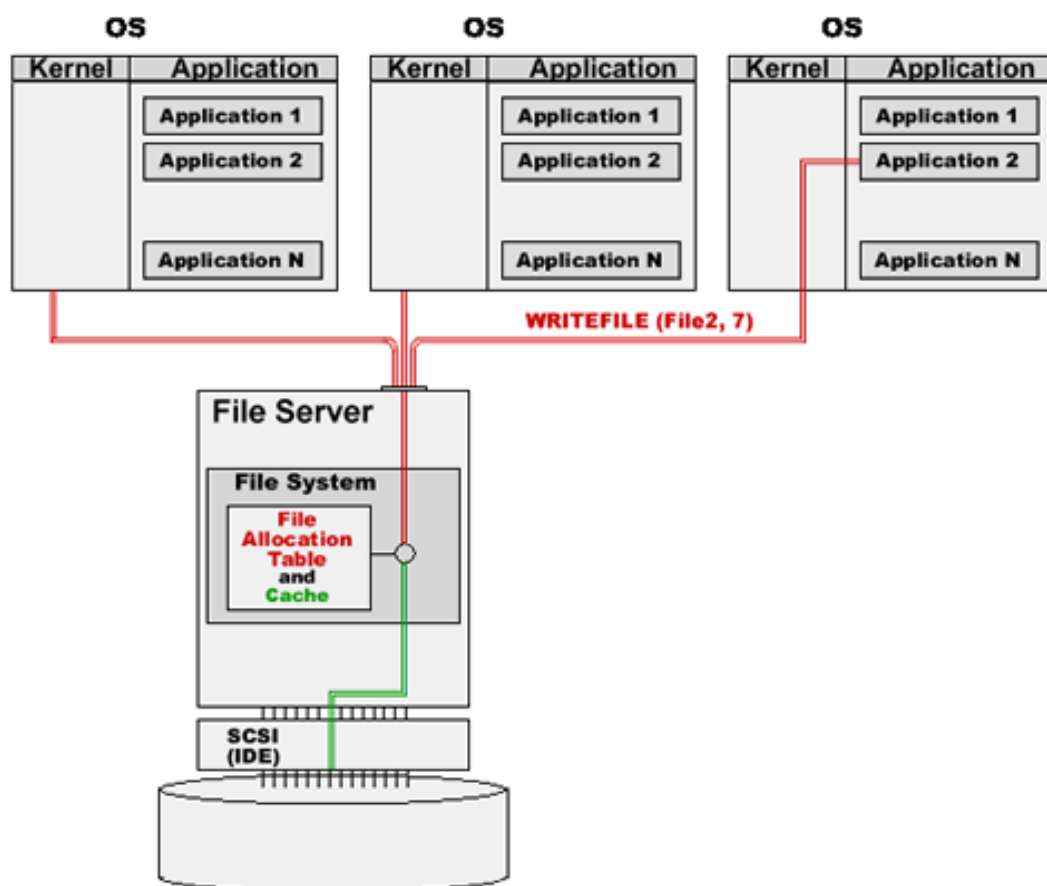
Step 10.



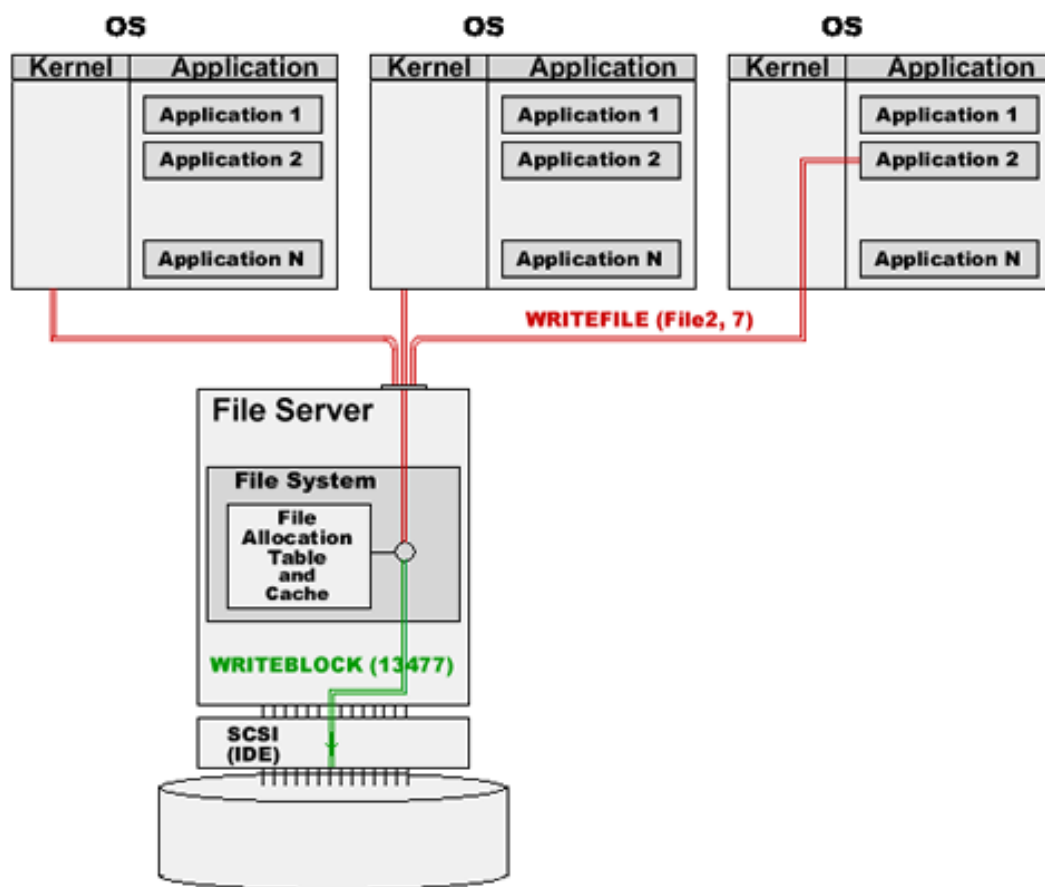
Step 11.



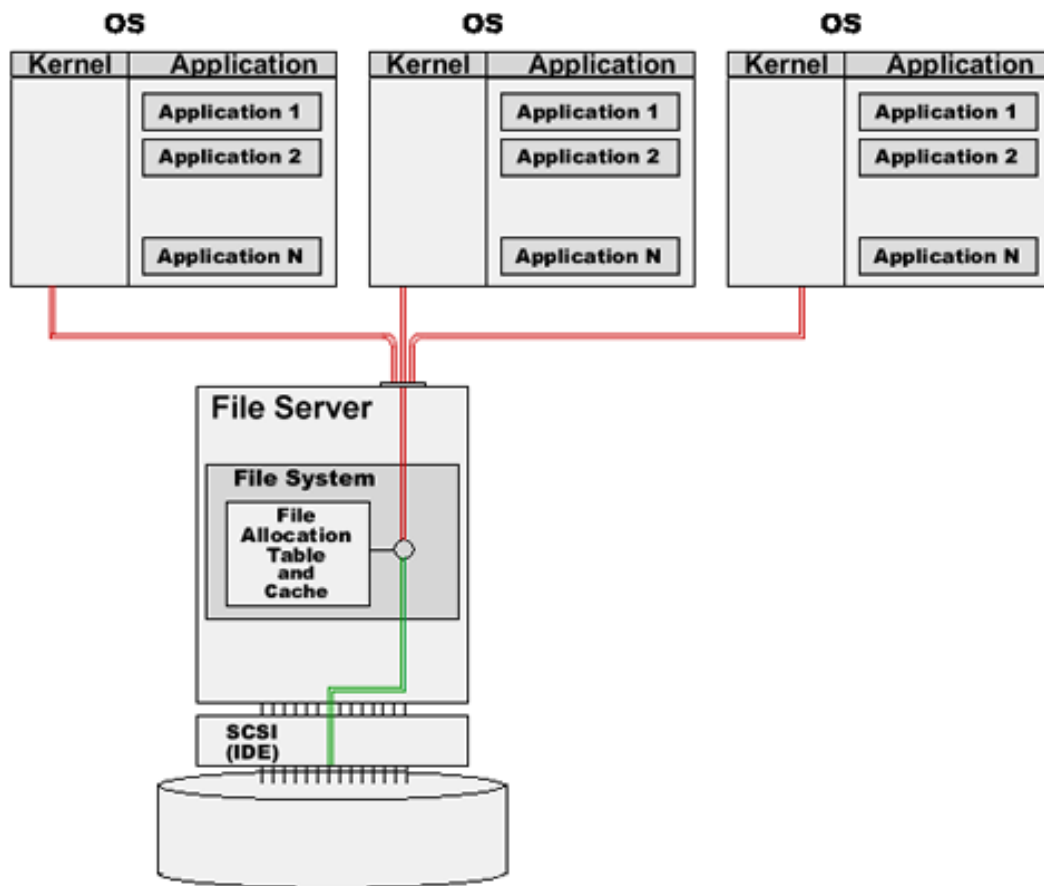
Step 12.



Step 13.



Step 14.



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.

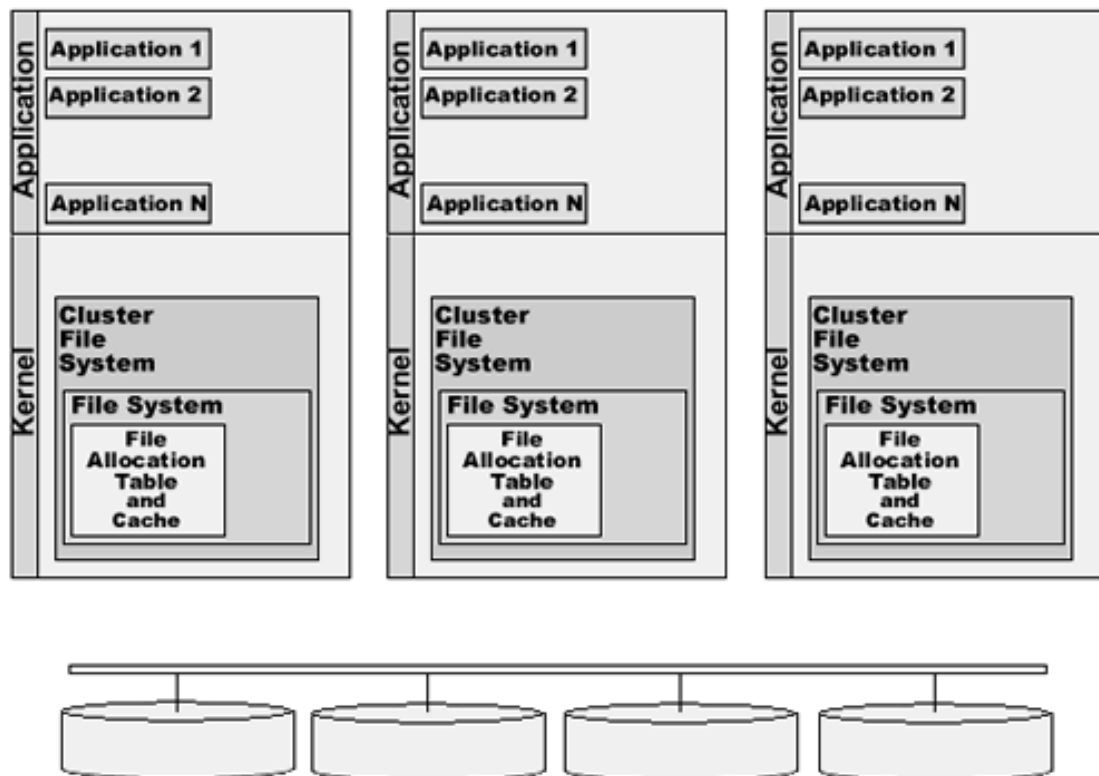
These problems make it impossible to use Shared Disks with regular File Systems as Shared File Systems. They can be used for fail-over systems or in any other configuration where only one computer is actually using the disk at any given time. The File System on computer Y starts to process the Shared Disk only when computer X has been shutdown, or stopped using the Shared Disk.

Cluster File System

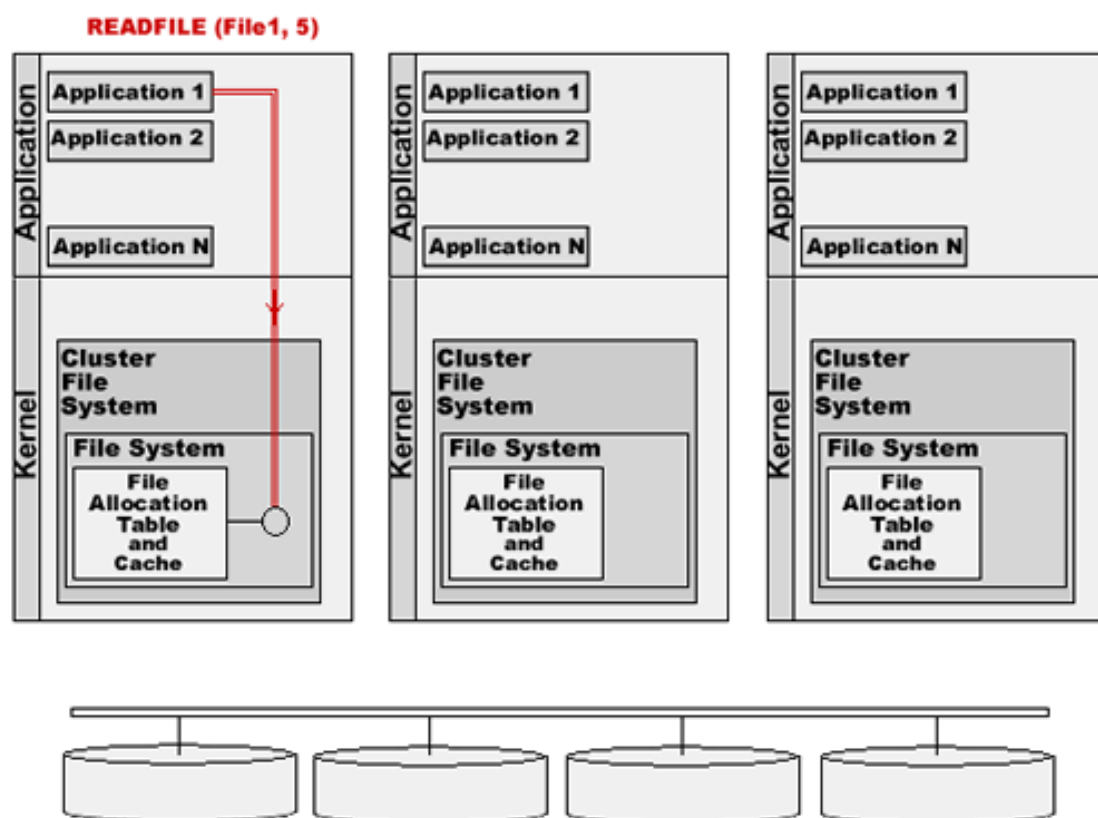
Cluster File Systems are software products designed to solve the problems outlined above. They allow you to build multi-computer systems with Shared Disks, solving the inconsistency problems.

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.

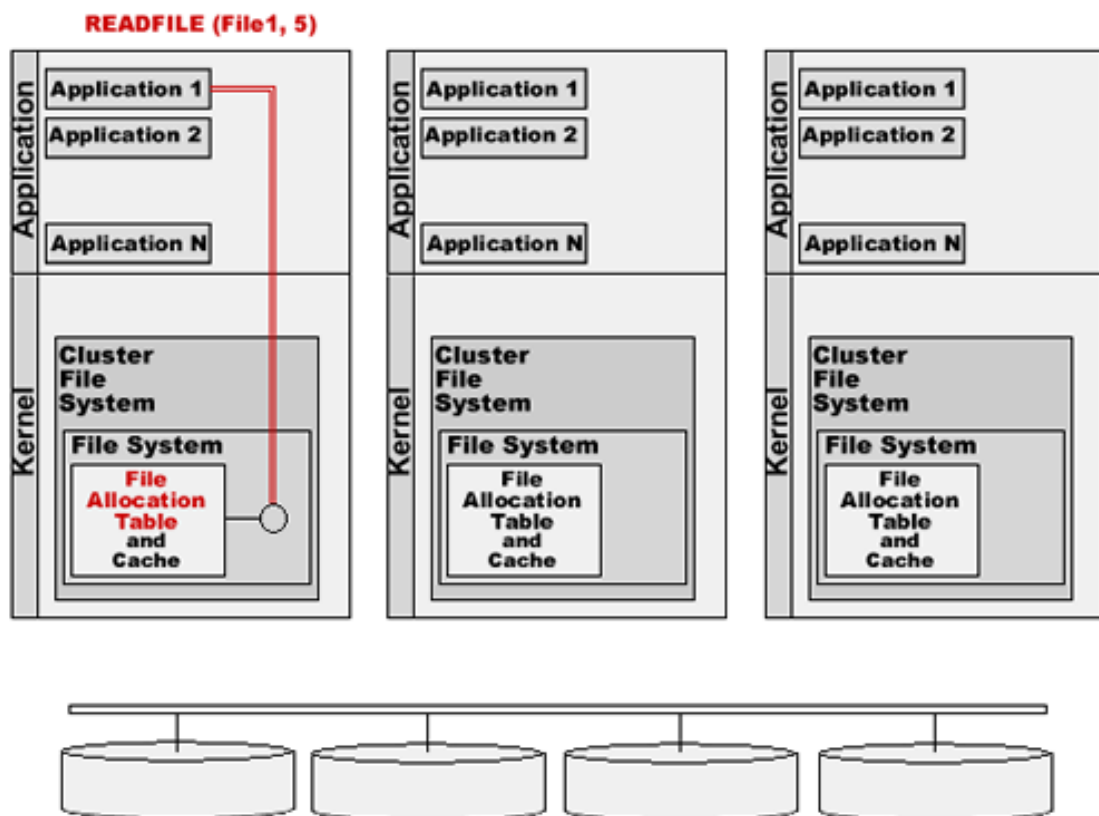
Step 1.



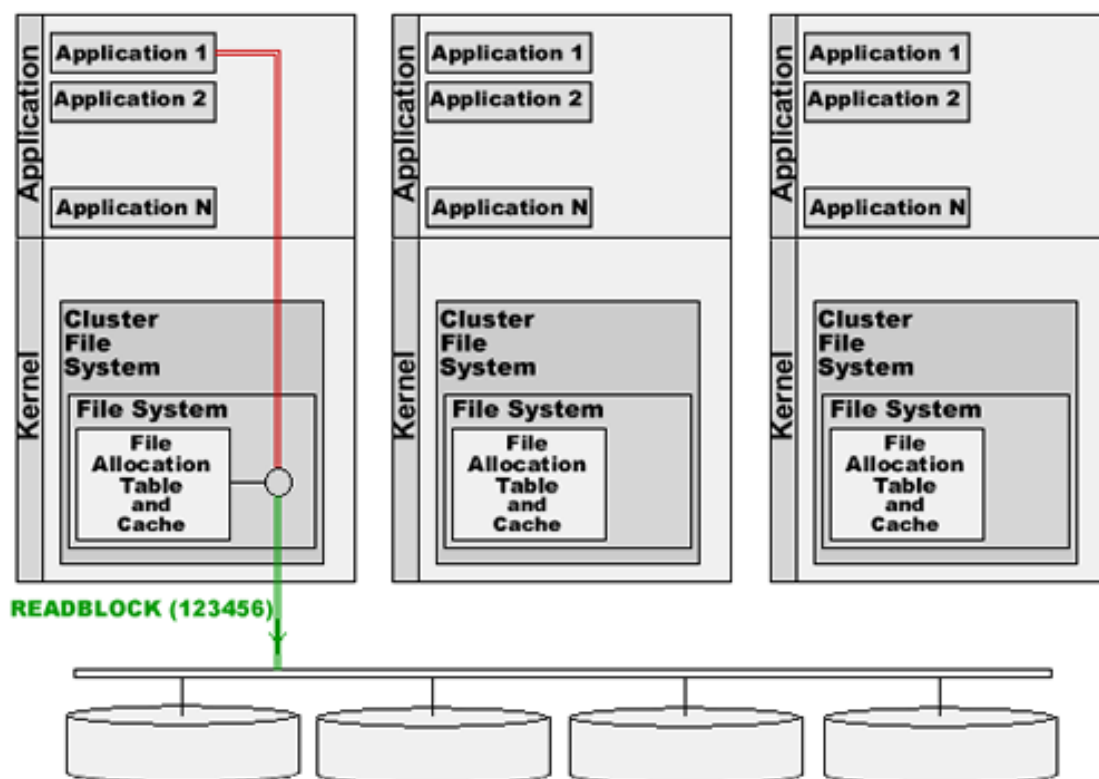
Step 2.



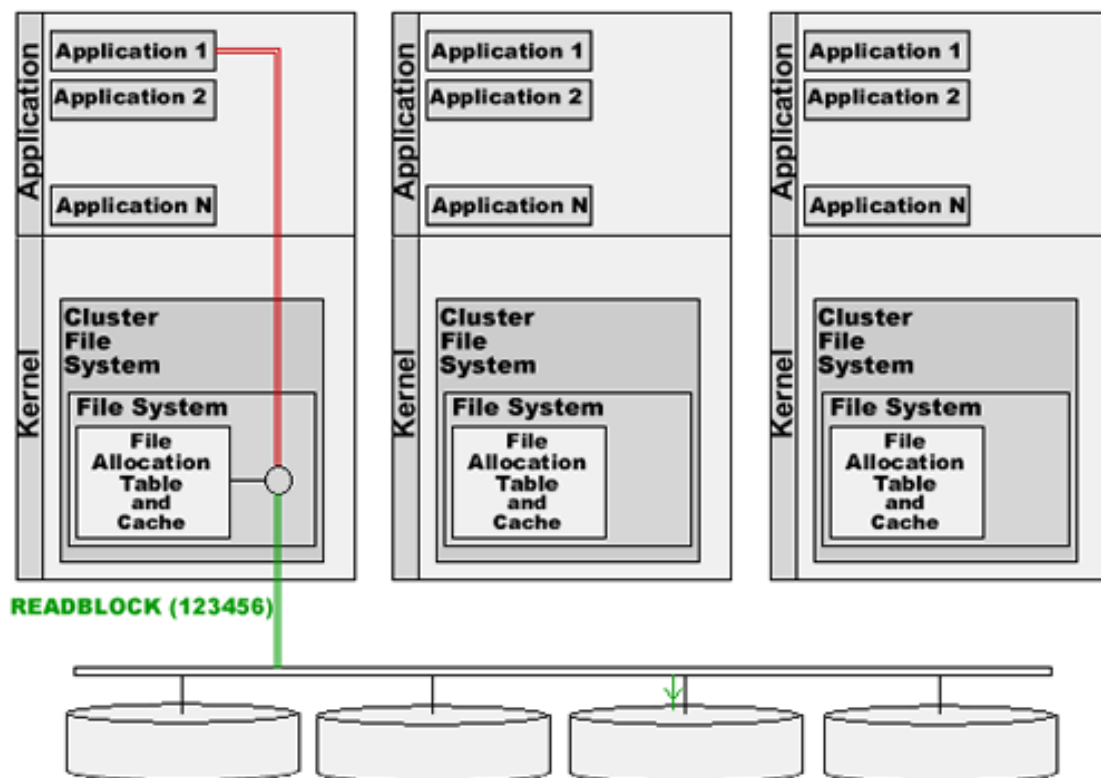
Step 3.



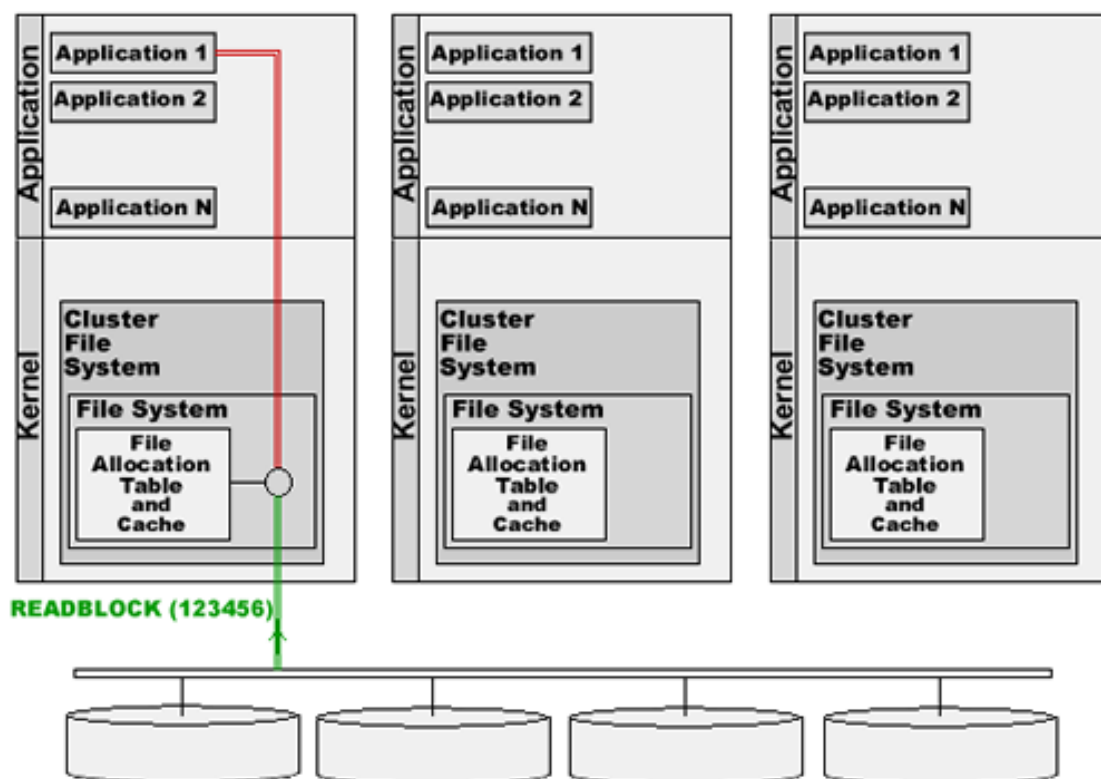
Step 4.



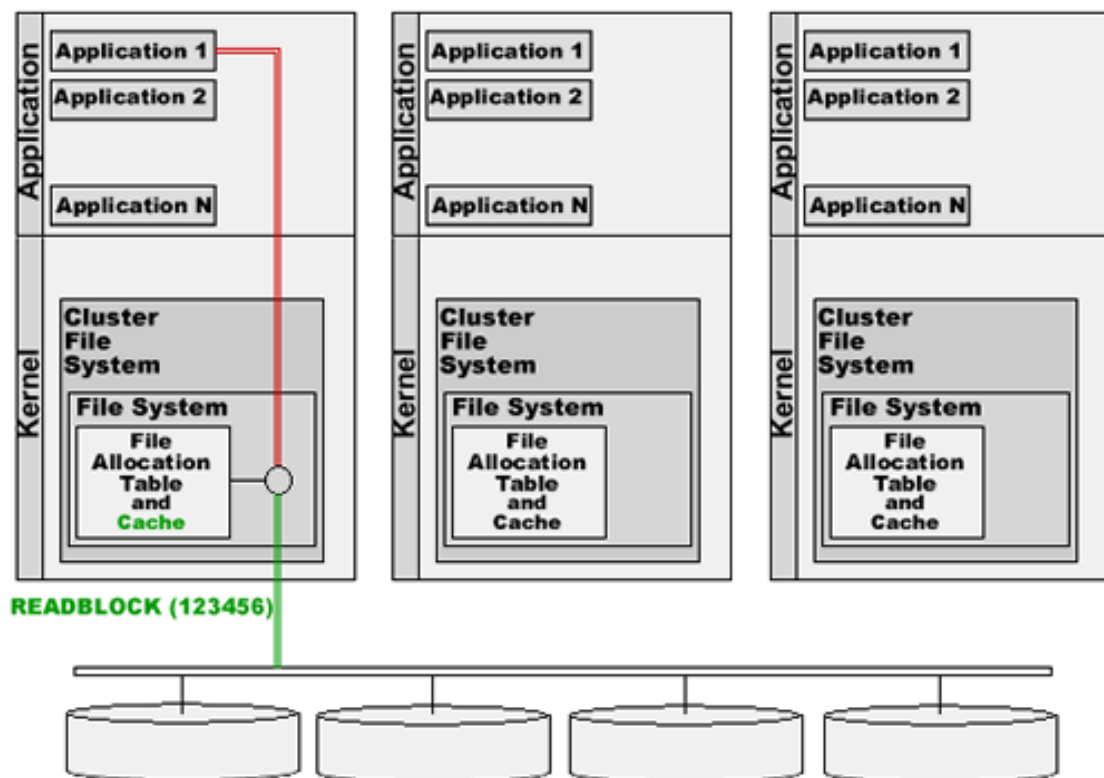
Step 5.



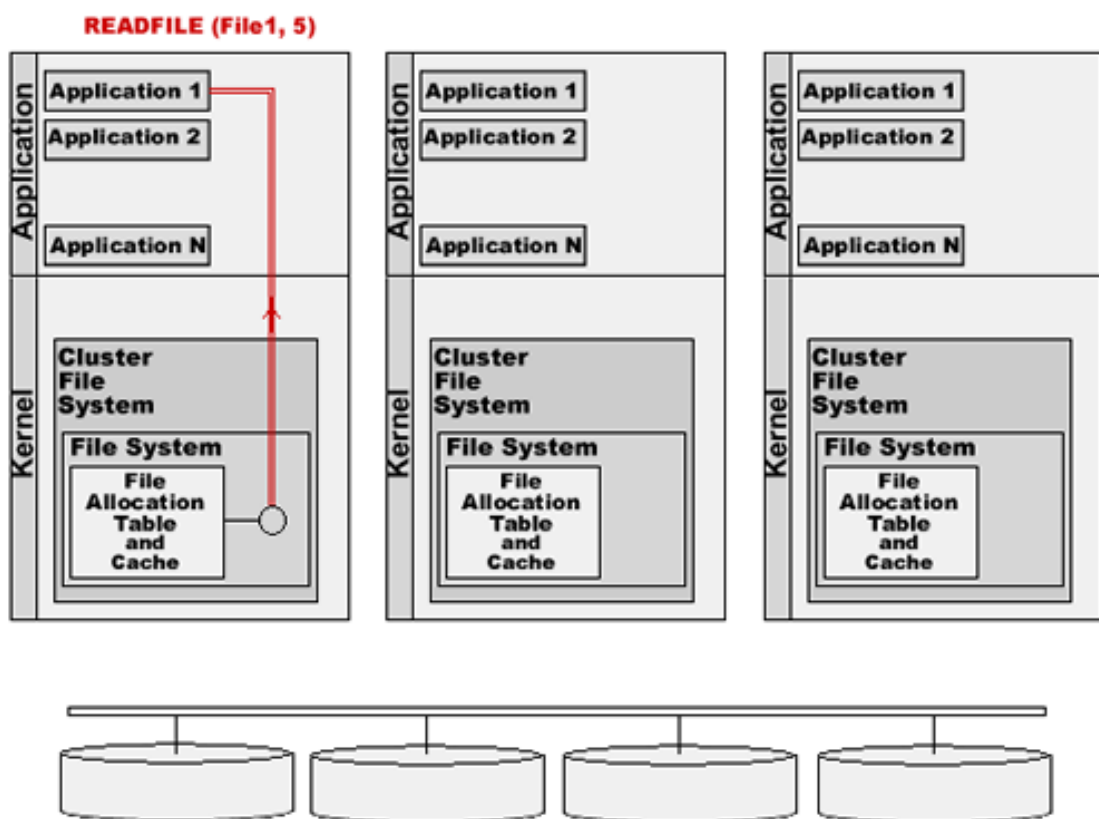
Step 6.



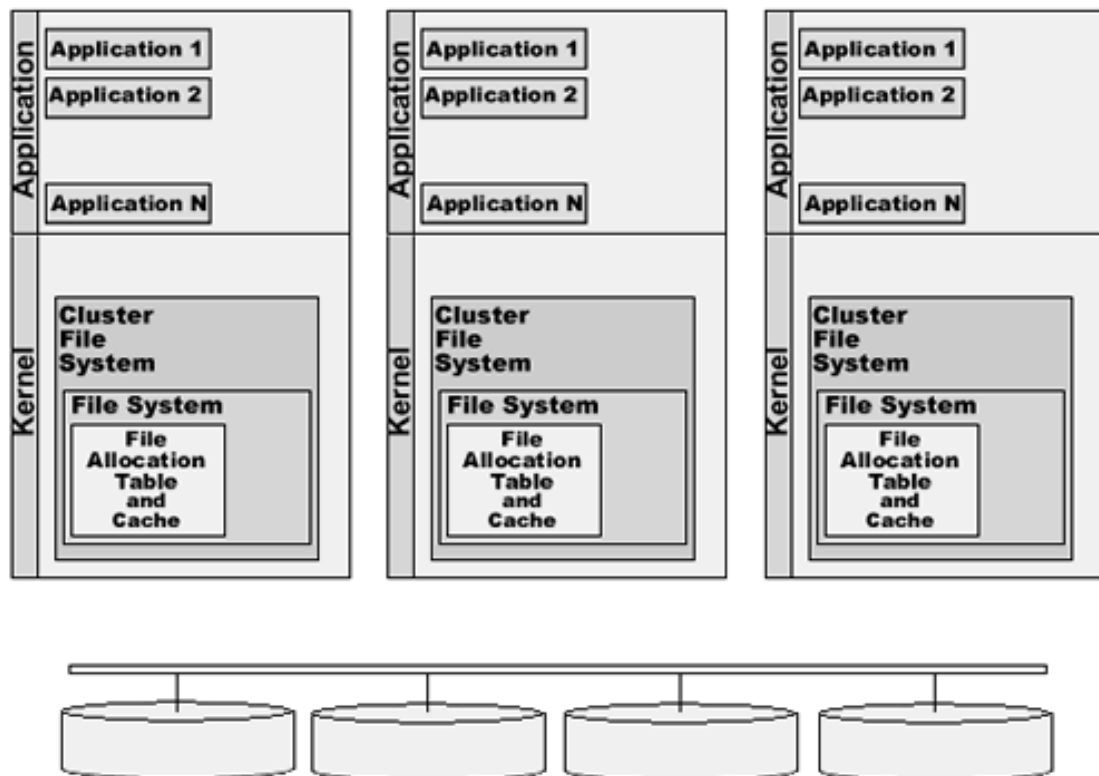
Step 7.



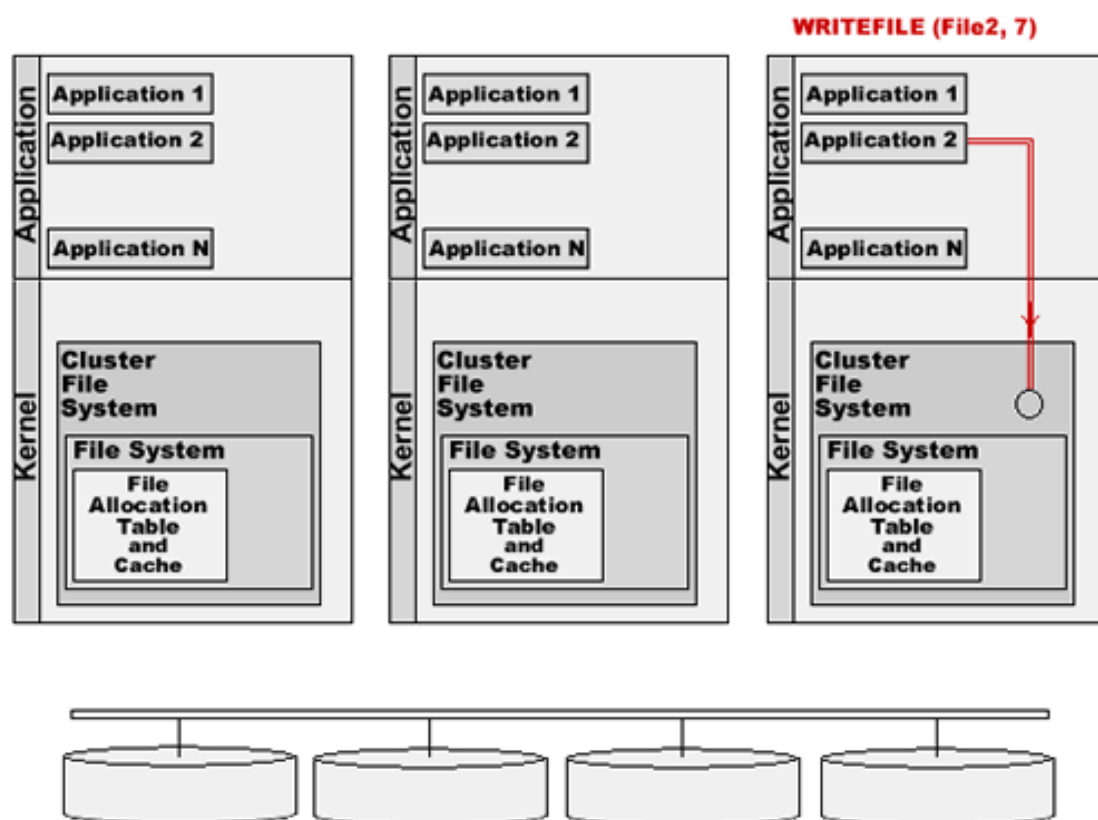
Step 8.



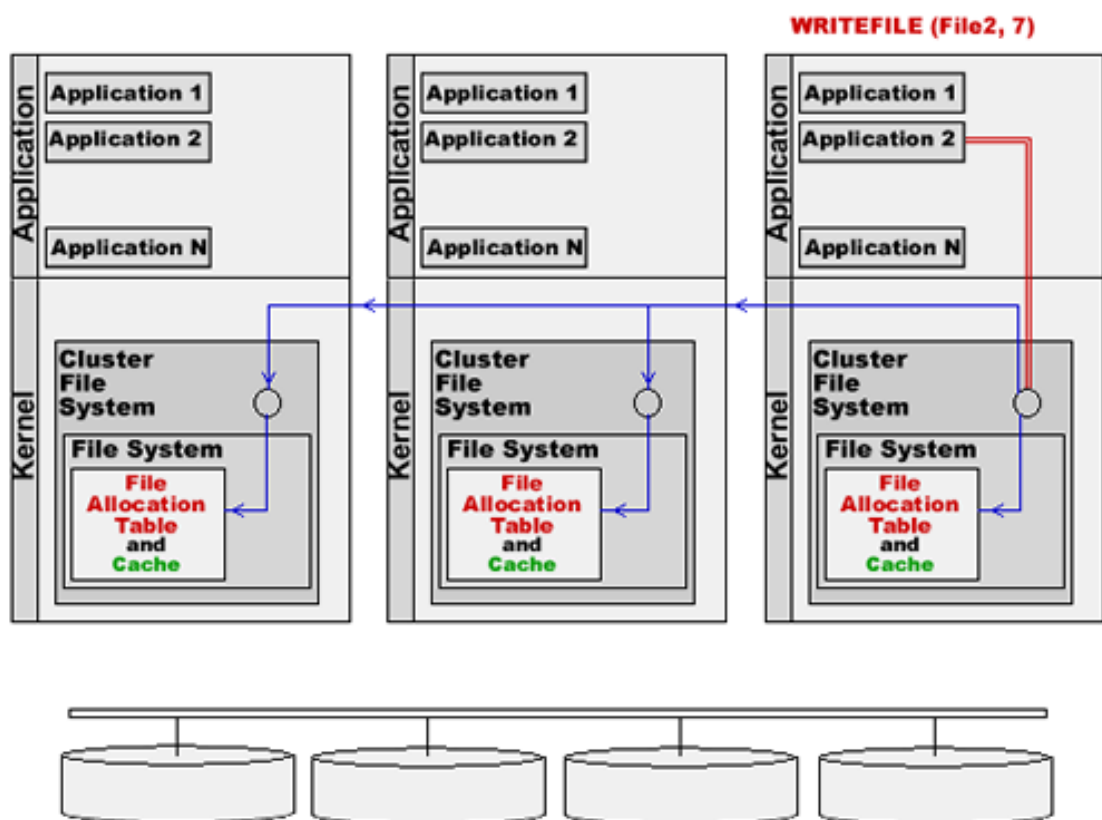
Step 9.



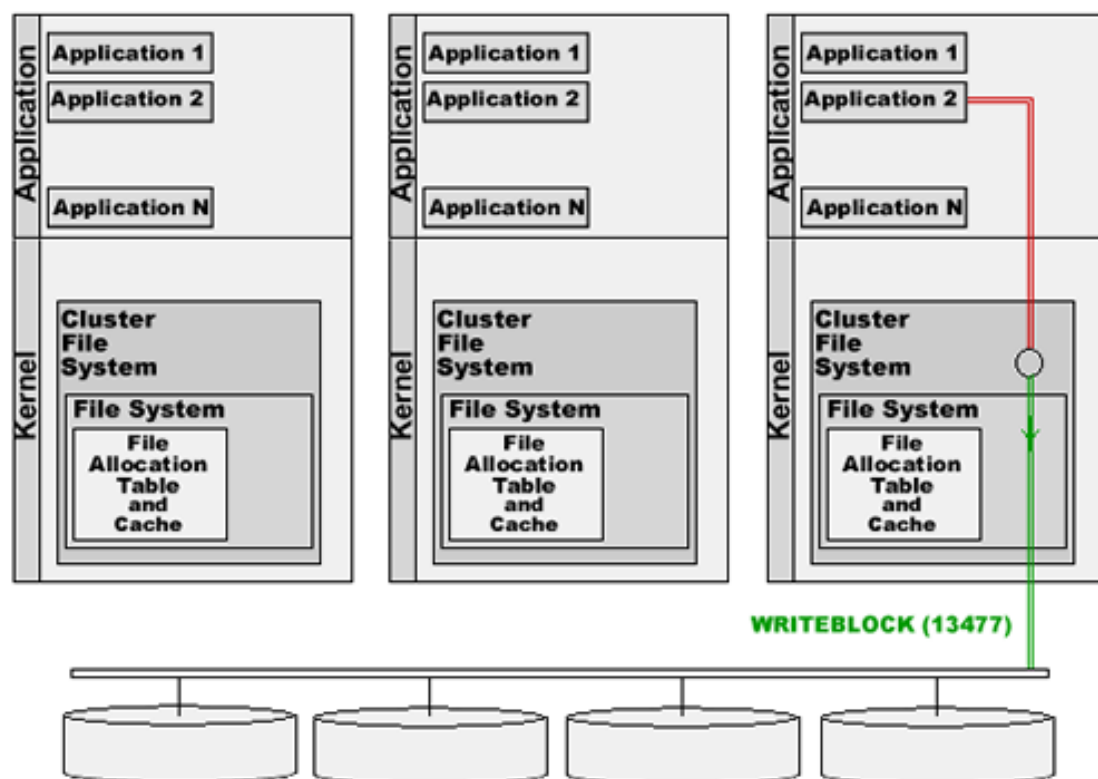
Step 10.



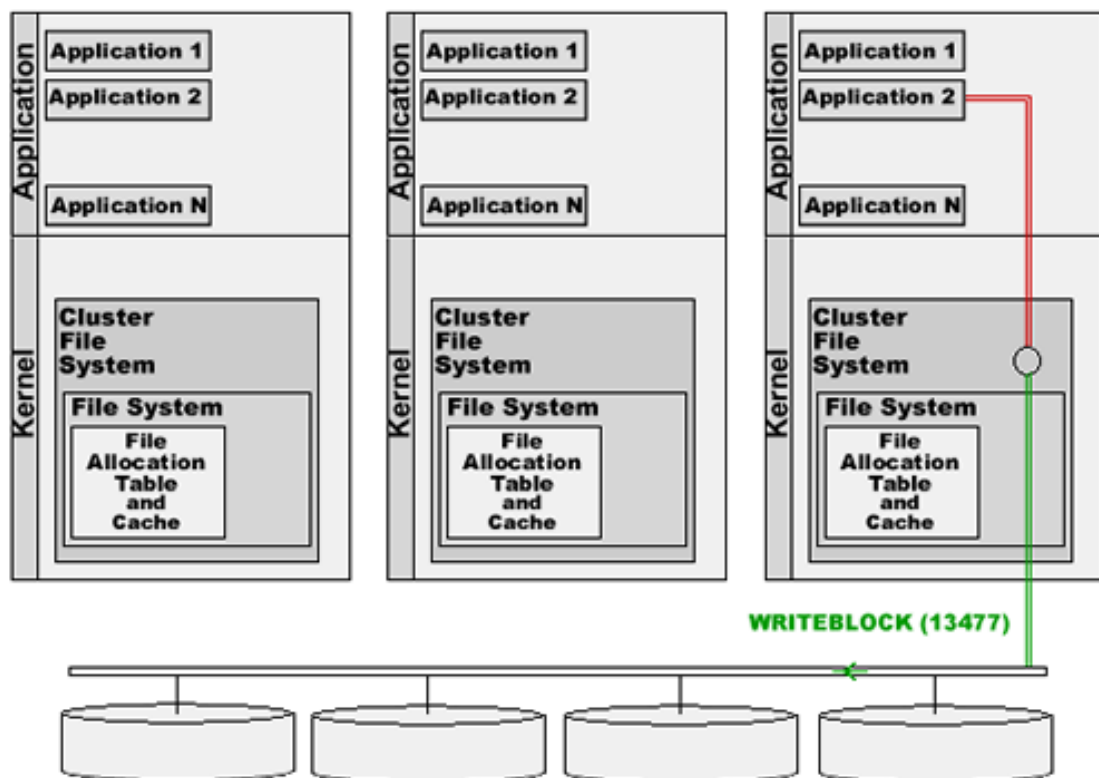
Step 11.



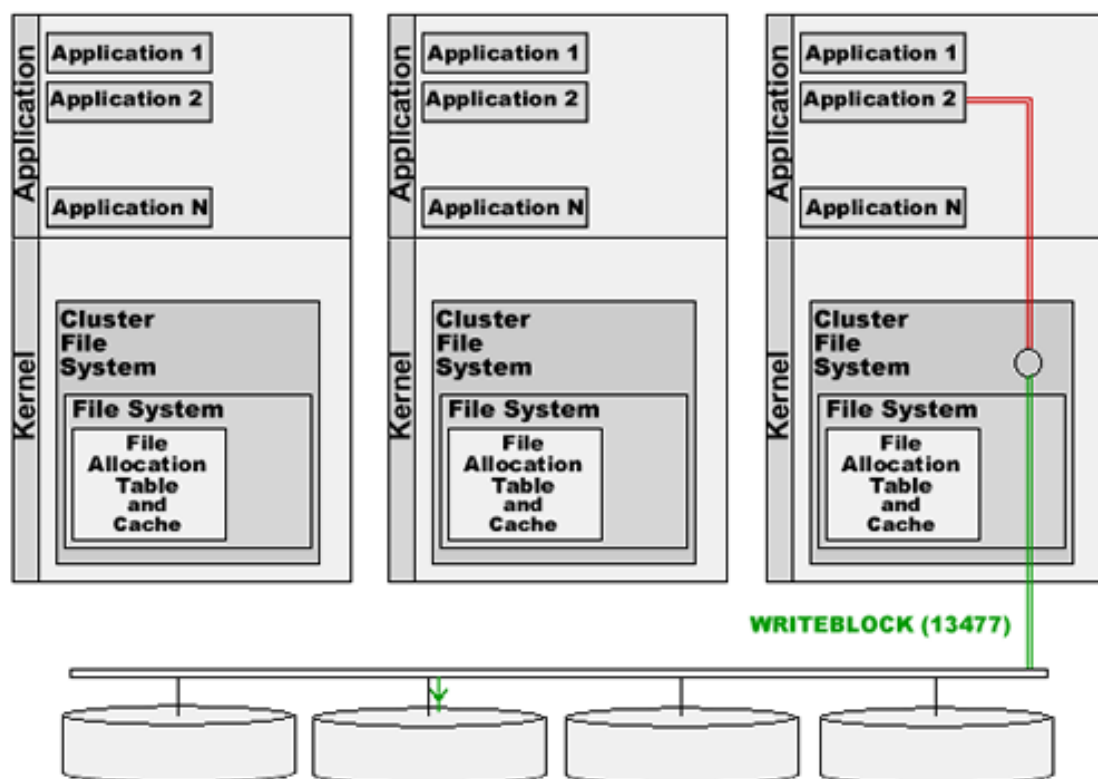
Step 12.



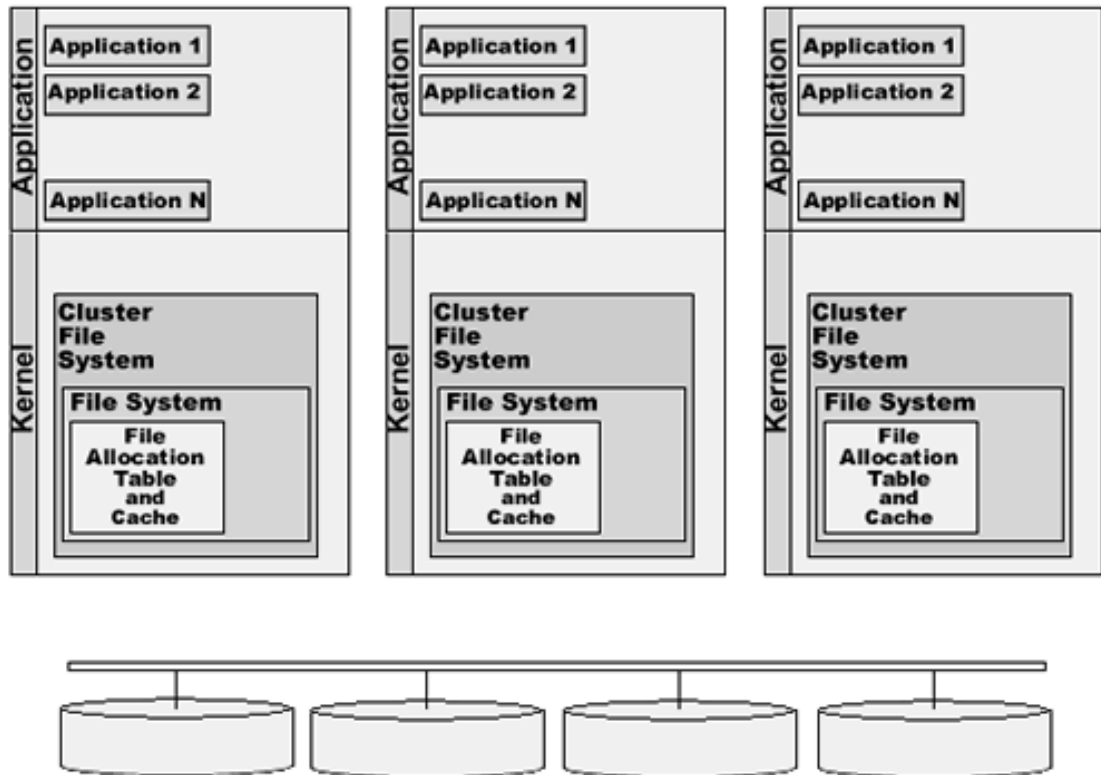
Step 13.



Step 14.



Step 15.



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



Cluster Transfer

This section explains how [Message Transfer](#) operations work in the CommuniGate Pro Cluster environment.

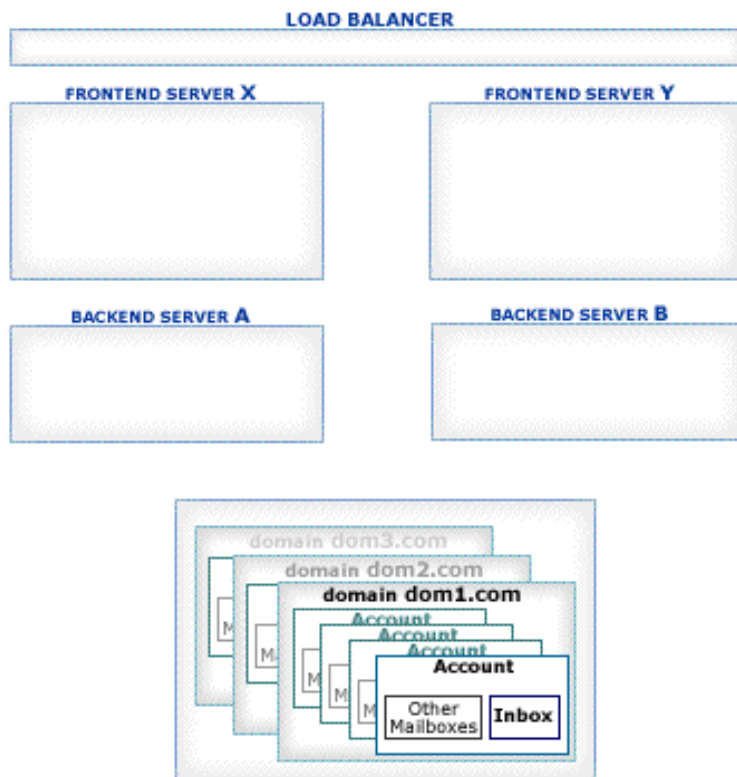
SMTP Relaying

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:

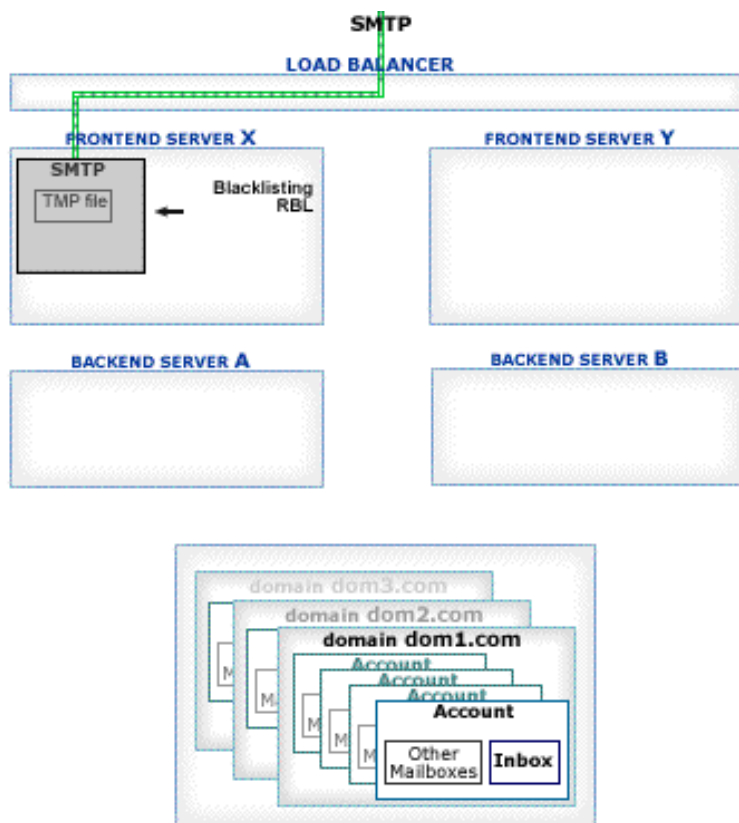
- authenticate the sender
- verify the sender and recipient addresses
- check the recipient status
- retrieve the sender limits

Received messages are enqueued. If a message is directed to an external address, it can be relayed by the same Frontend:

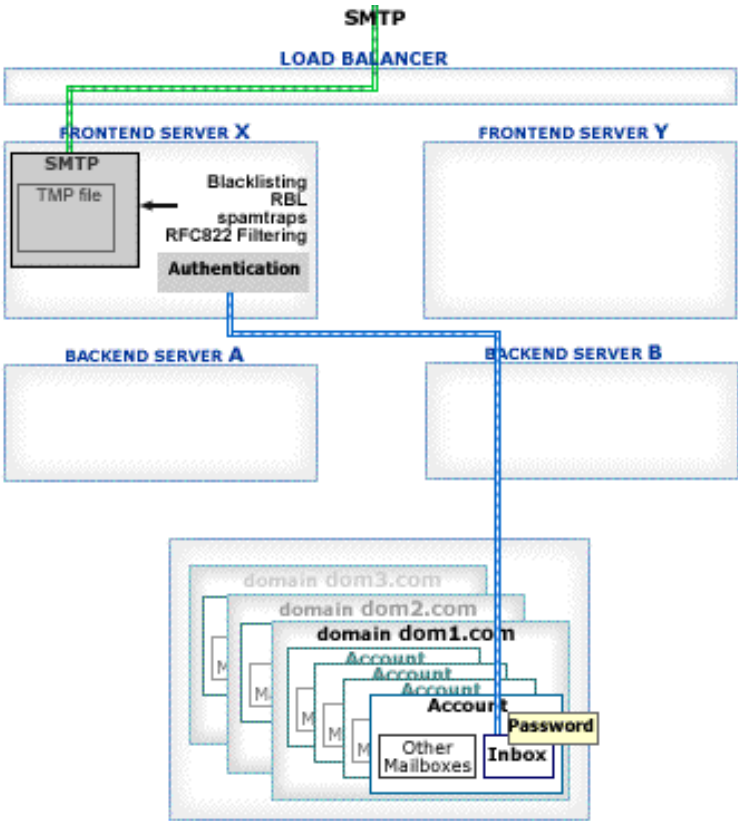
Step 1.



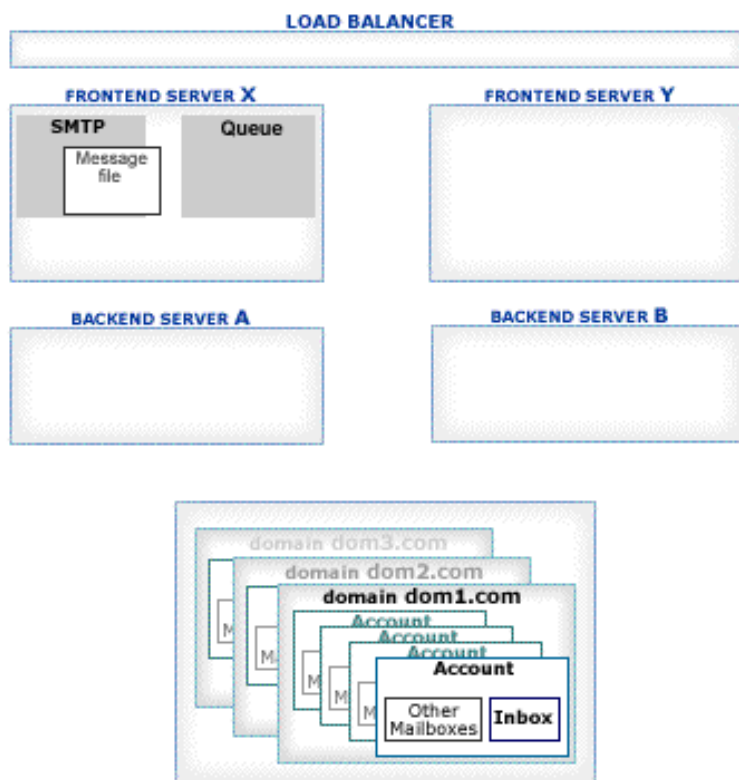
Step 2.



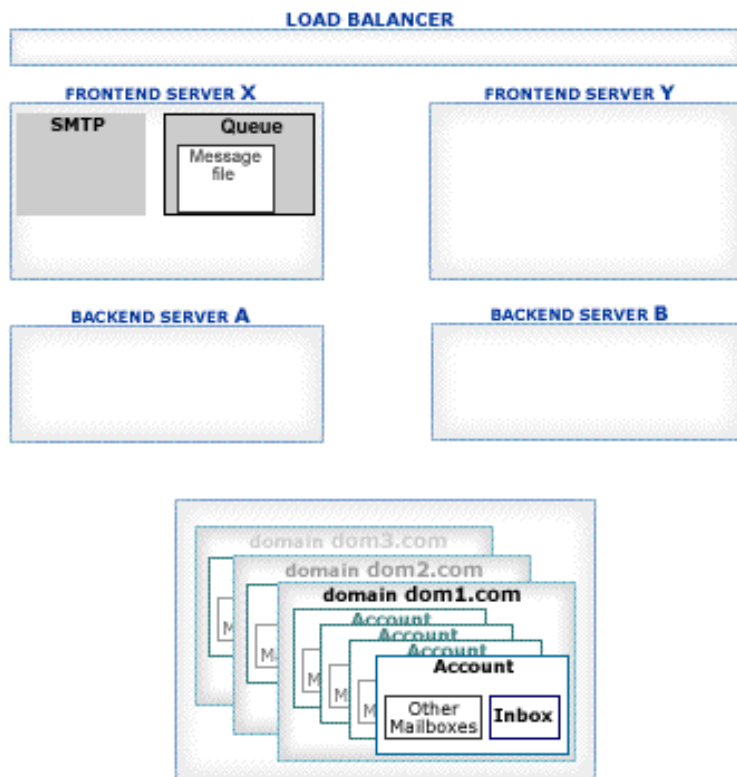
Step 3.



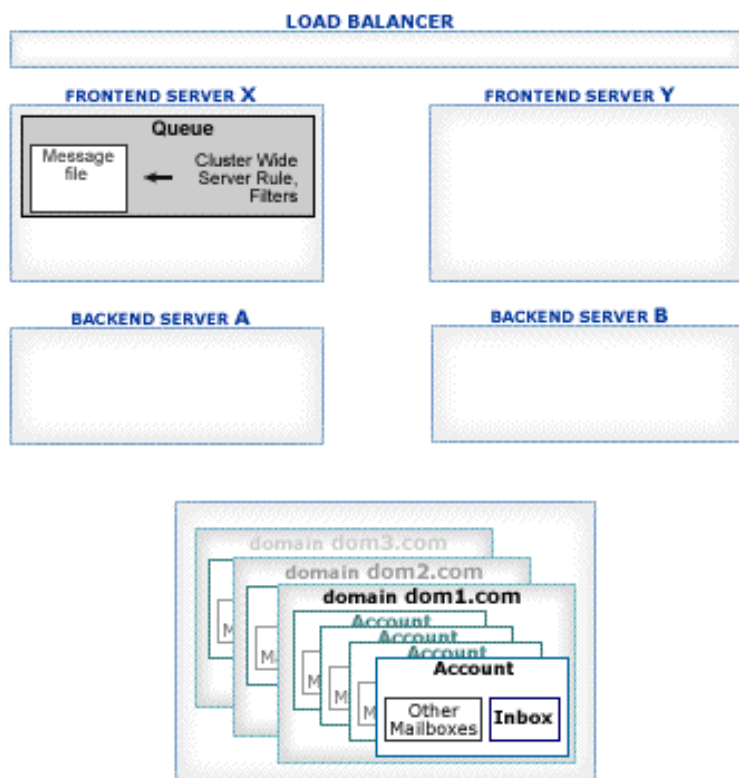
Step 4.



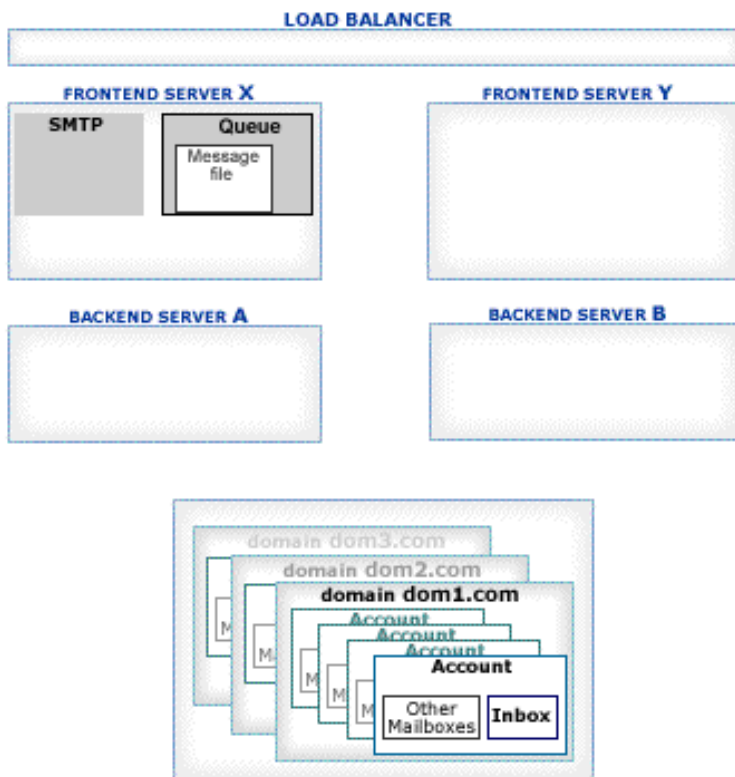
Step 5.



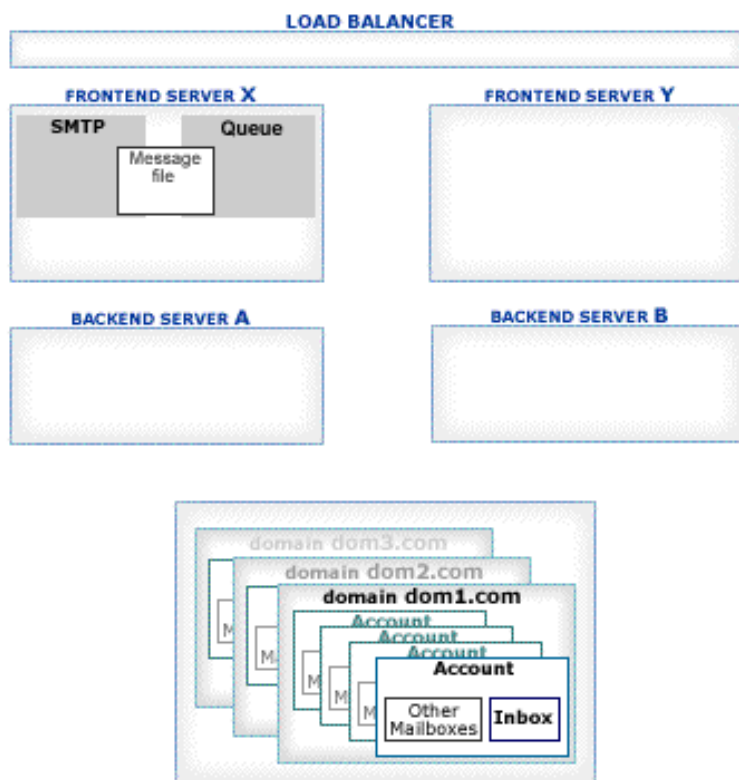
Step 6.



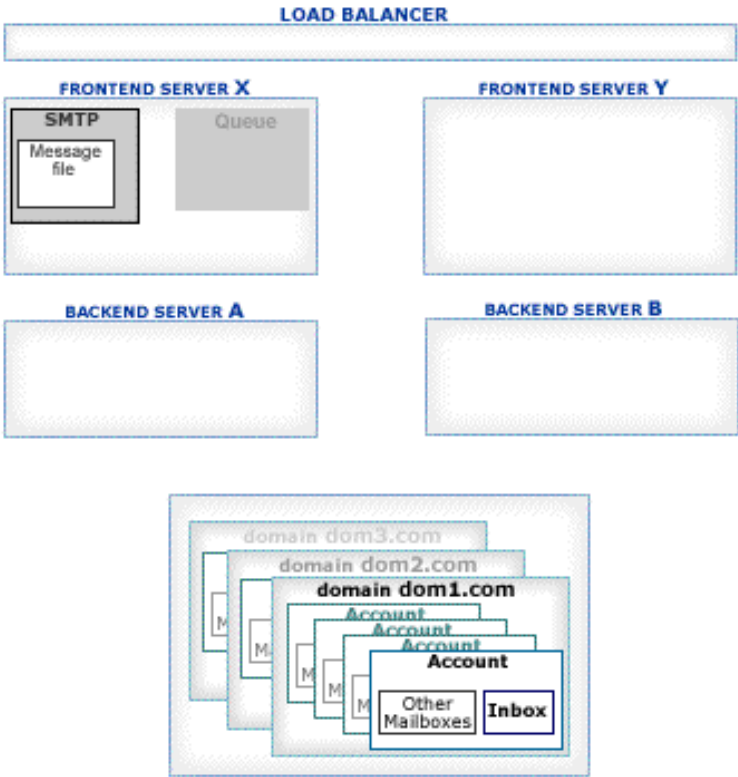
Step 7.



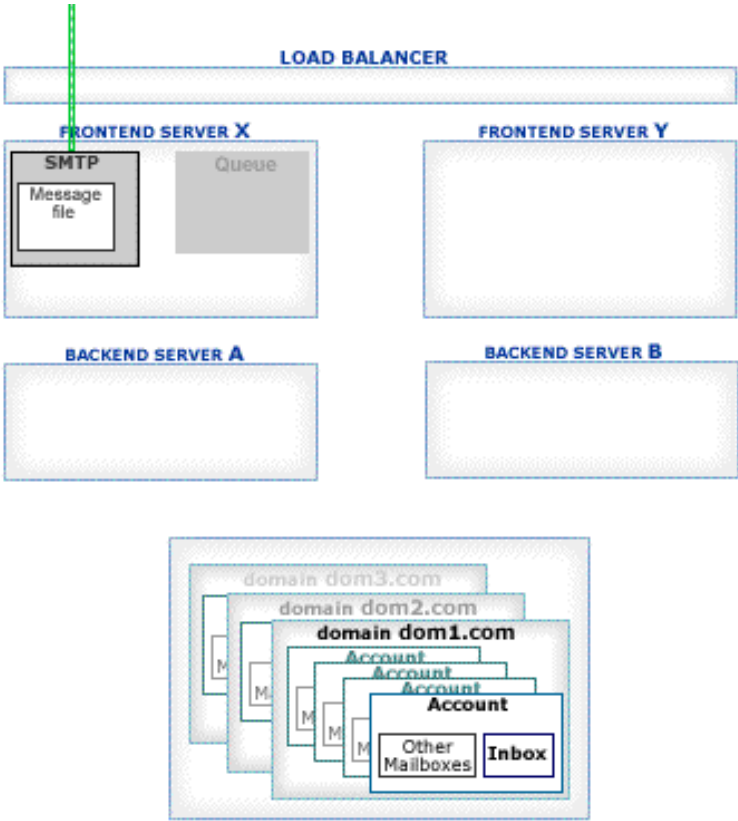
Step 8.



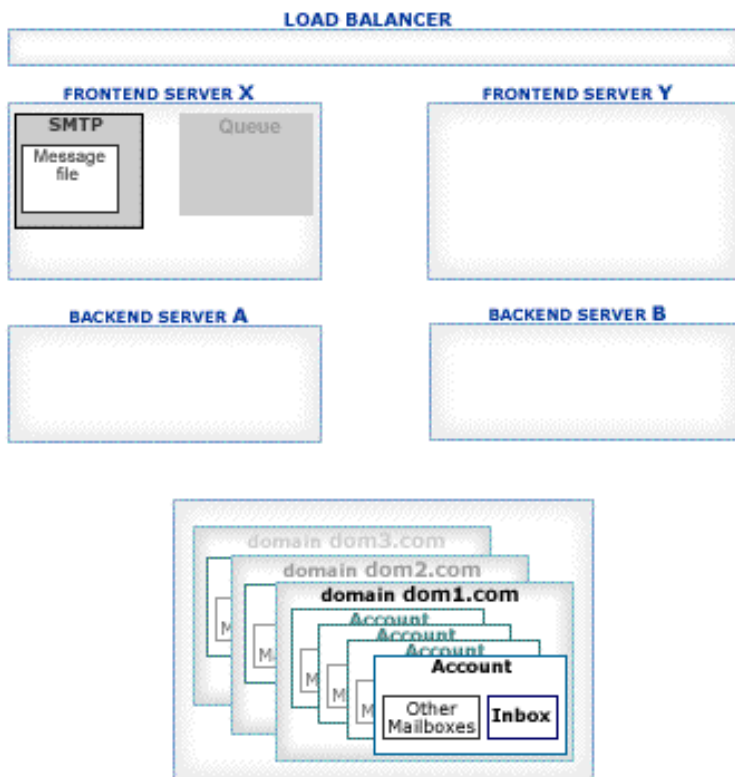
Step 9.



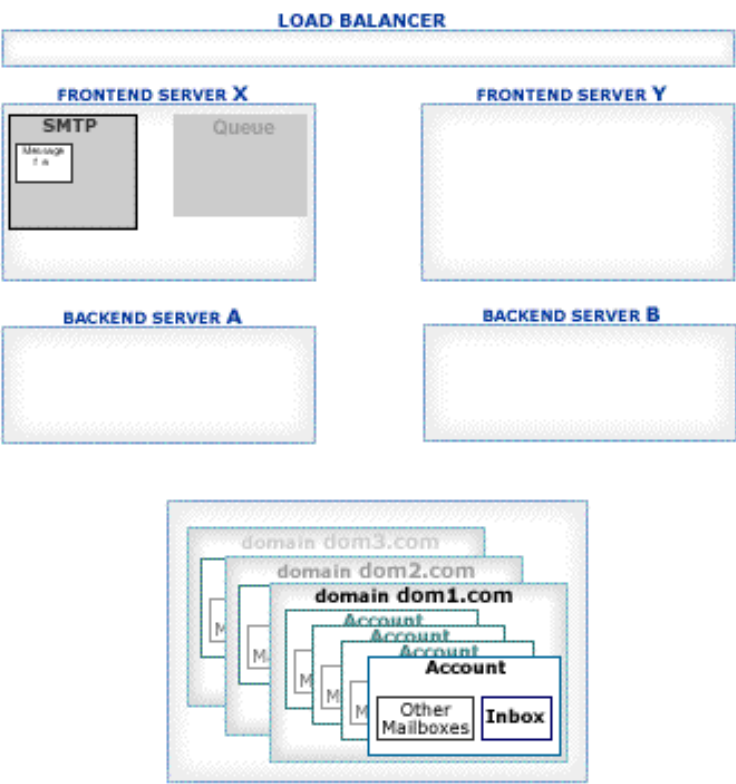
Step 10.



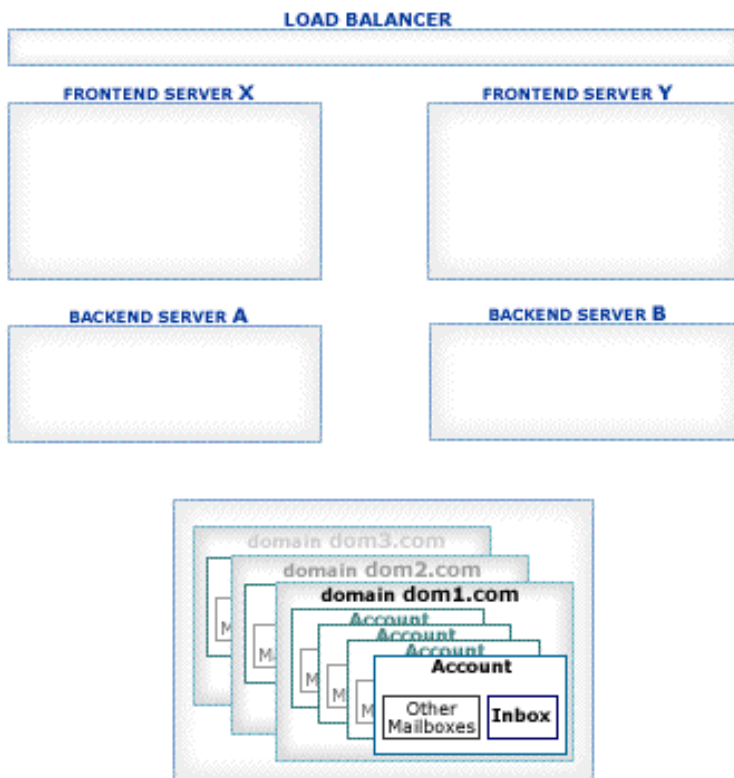
Step 11.



Step 12.



Step 13.



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.

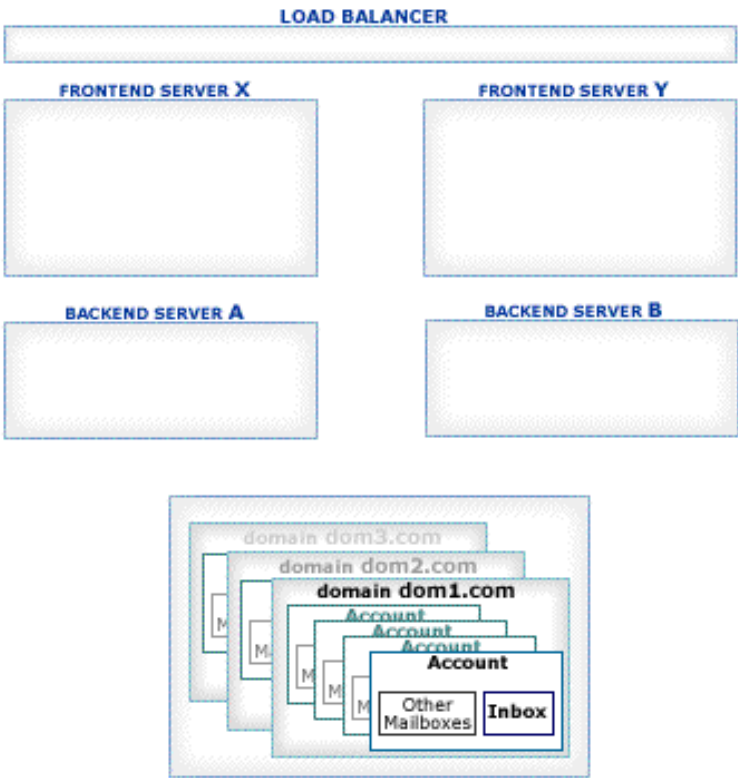
This situation can happen when a message is enqueued on a Frontend Server (Frontend Servers cannot directly open any Account in Shared Domains), or a message is enqueued on a Backend Server that either does not host the target Account (in a Static Cluster), or it cannot open it, because the Account is opened by some other Back-

end Server (in a Dynamic Cluster).

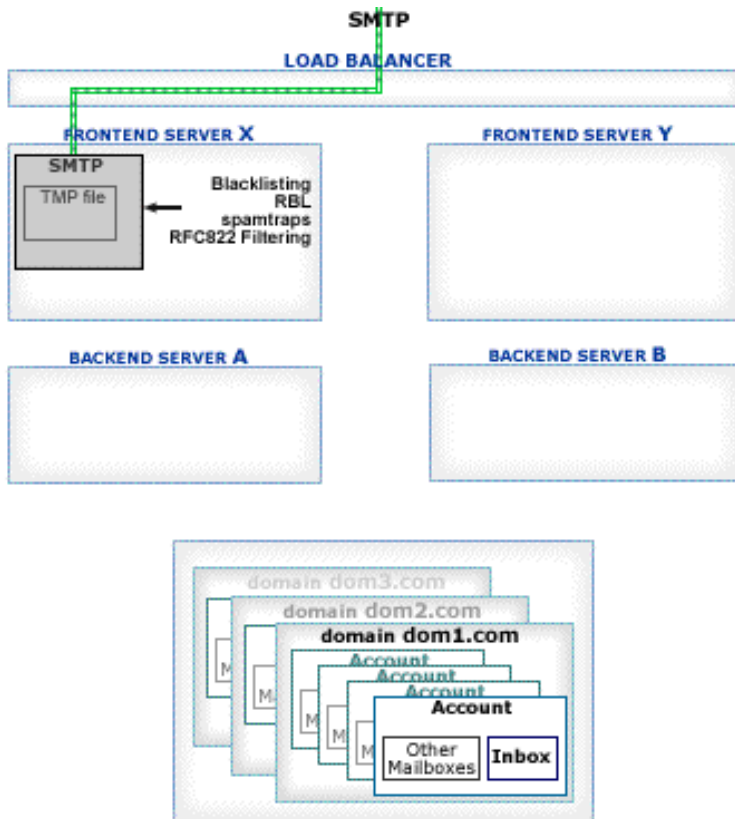
To solve the problem, the [Local Delivery](#) module uses a Delivery channel connection to the proper Backend Server and passes the message there. The receiving Backend immediately opens the target Account, applies its Account-Level Rule and stores the transferred message. This Backend does not enqueue 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).

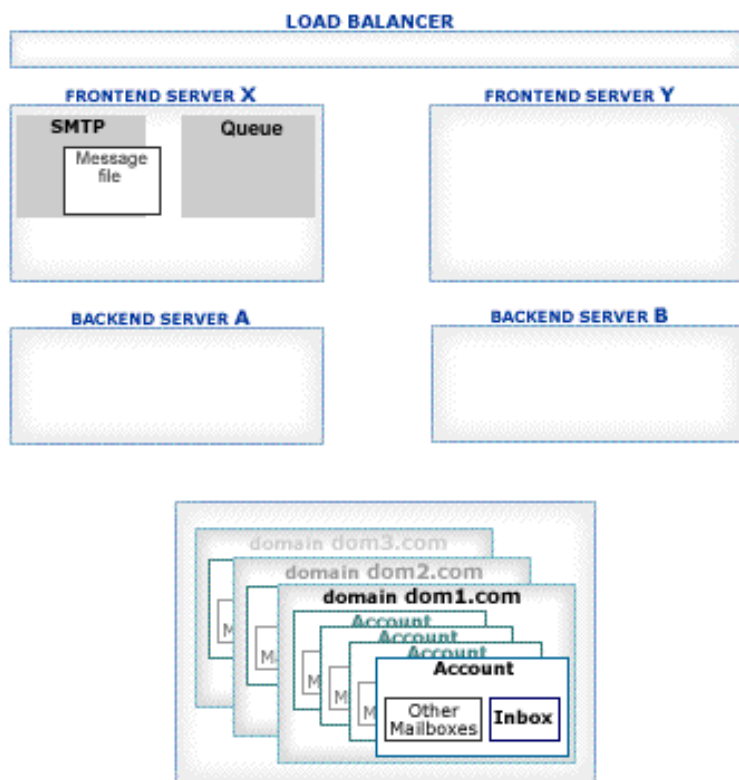
Step 1.



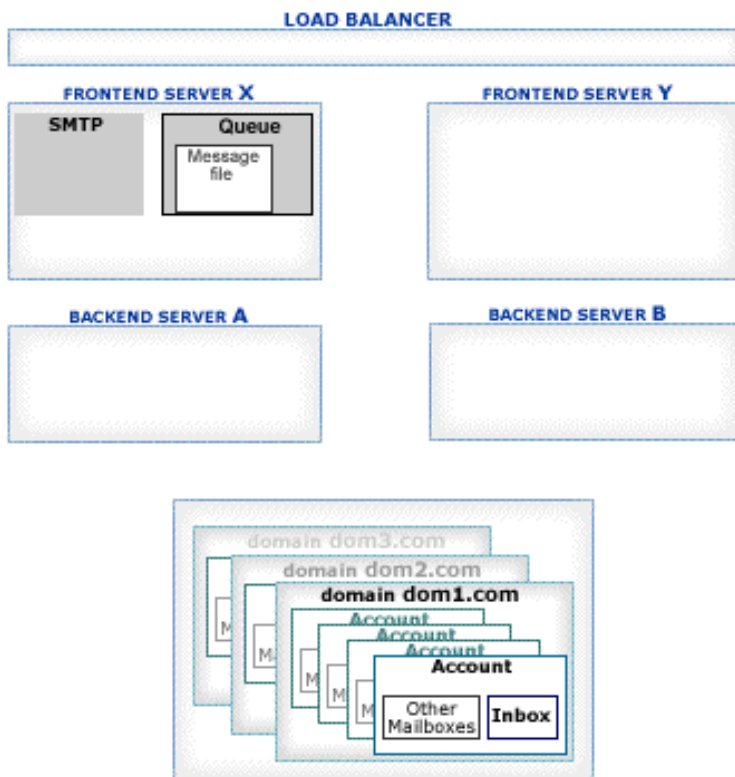
Step 2.



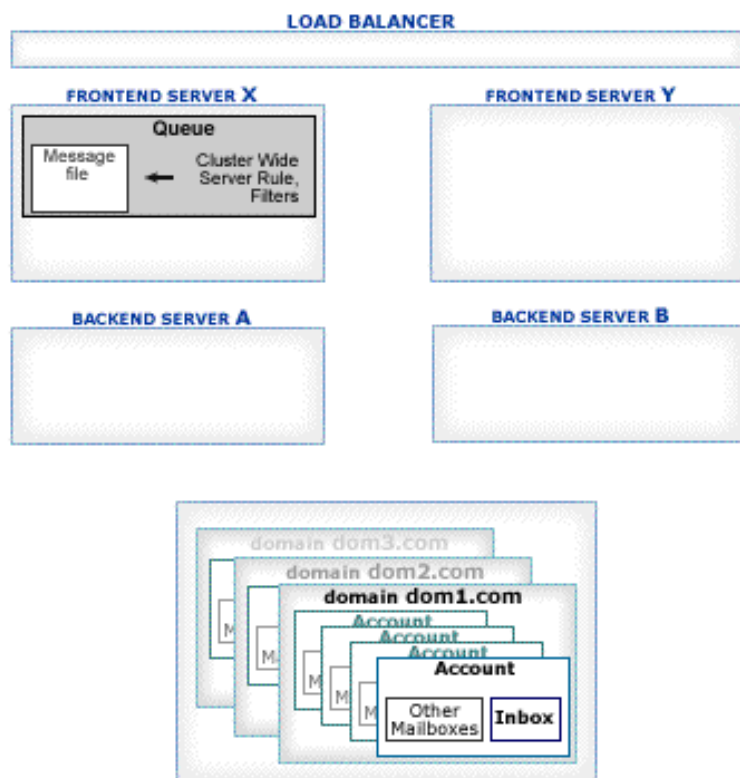
Step 3.



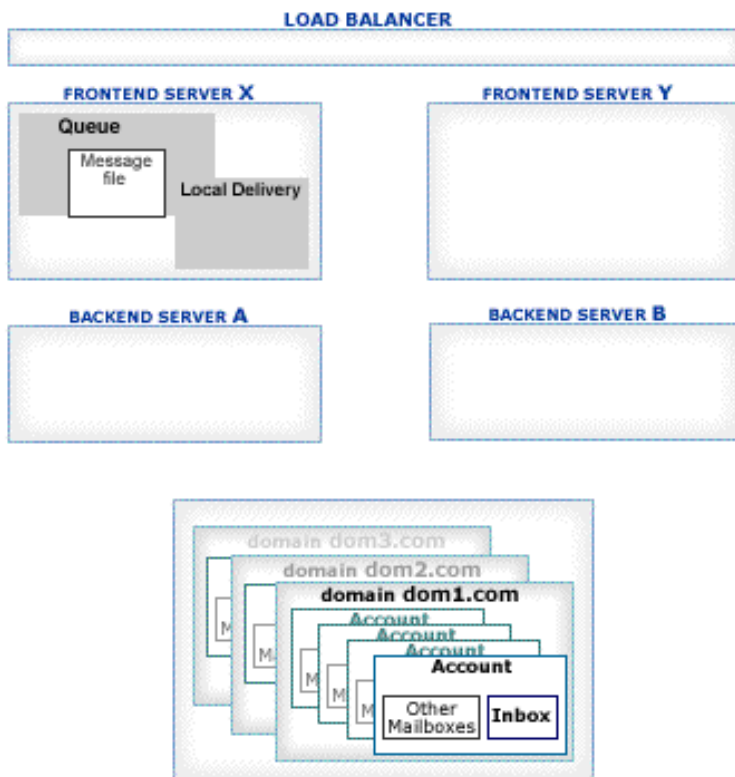
Step 4.



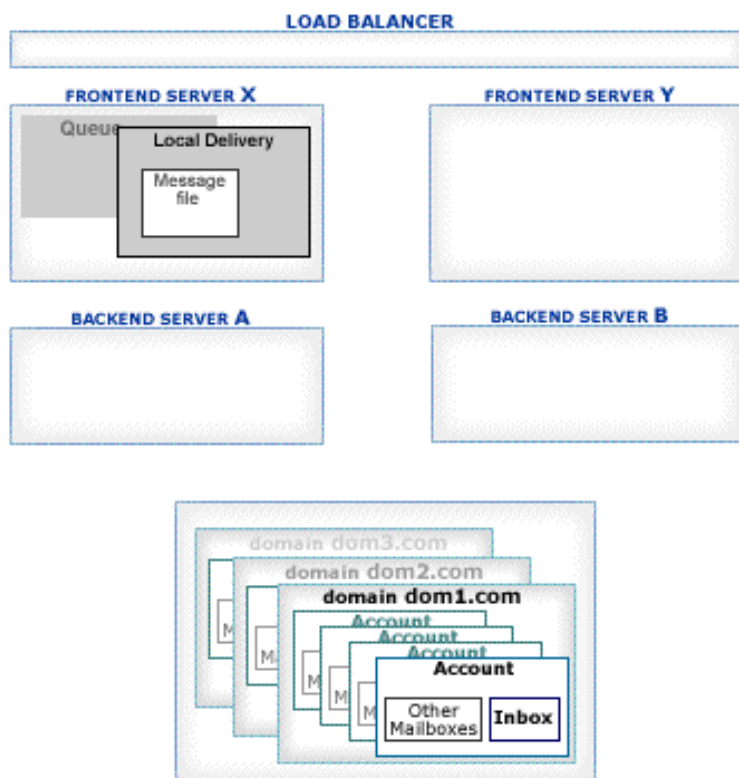
Step 5.



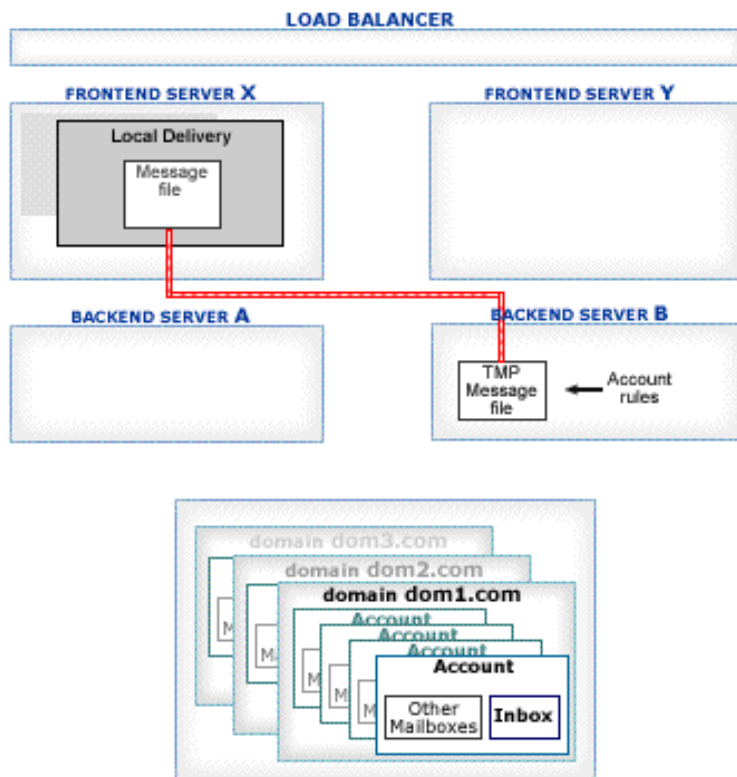
Step 6.



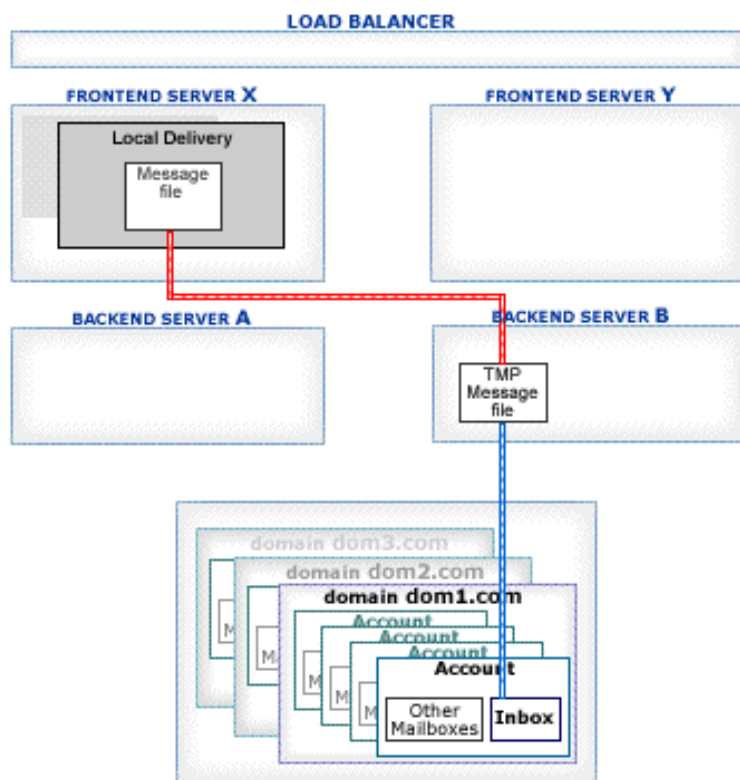
Step 7.



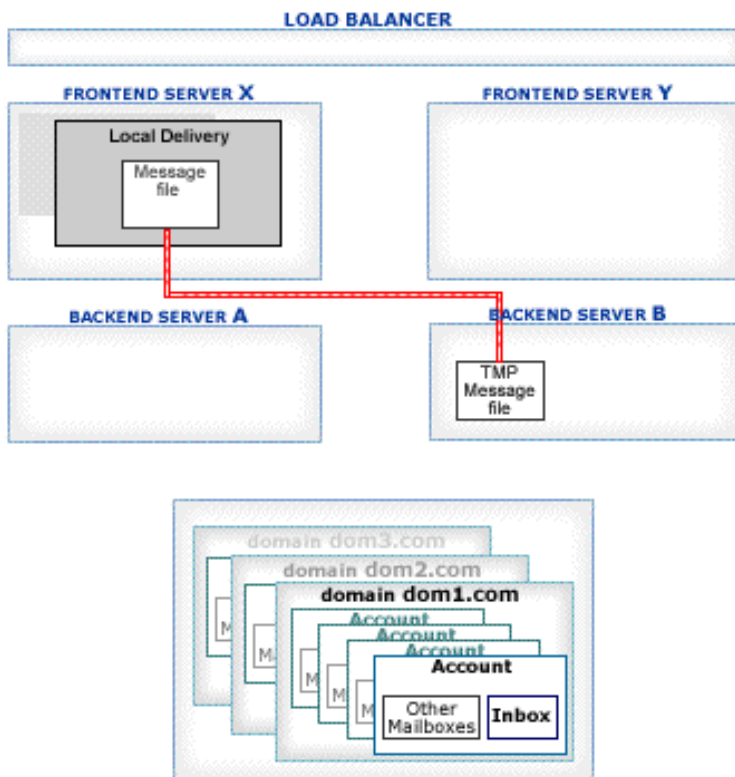
Step 8.



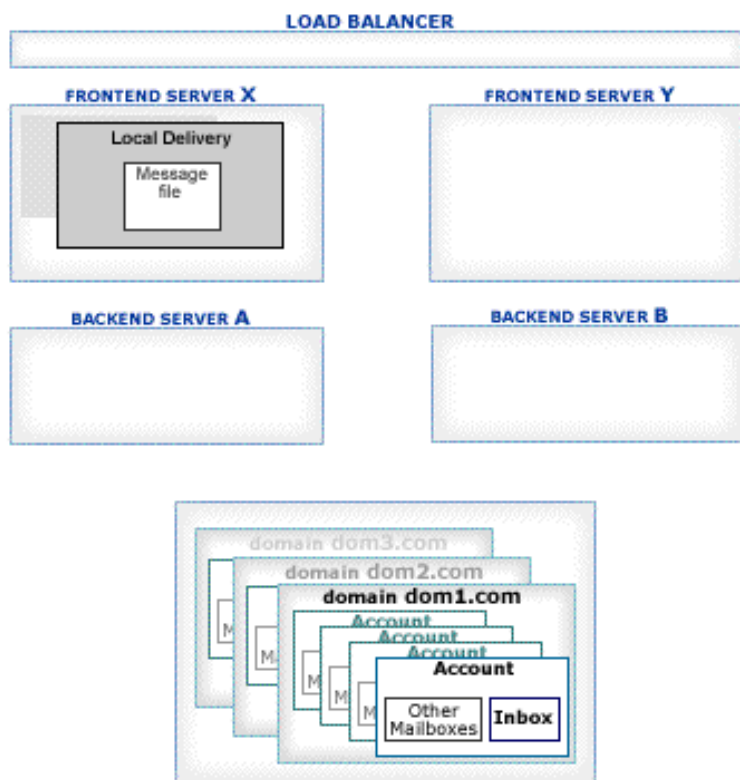
Step 9.



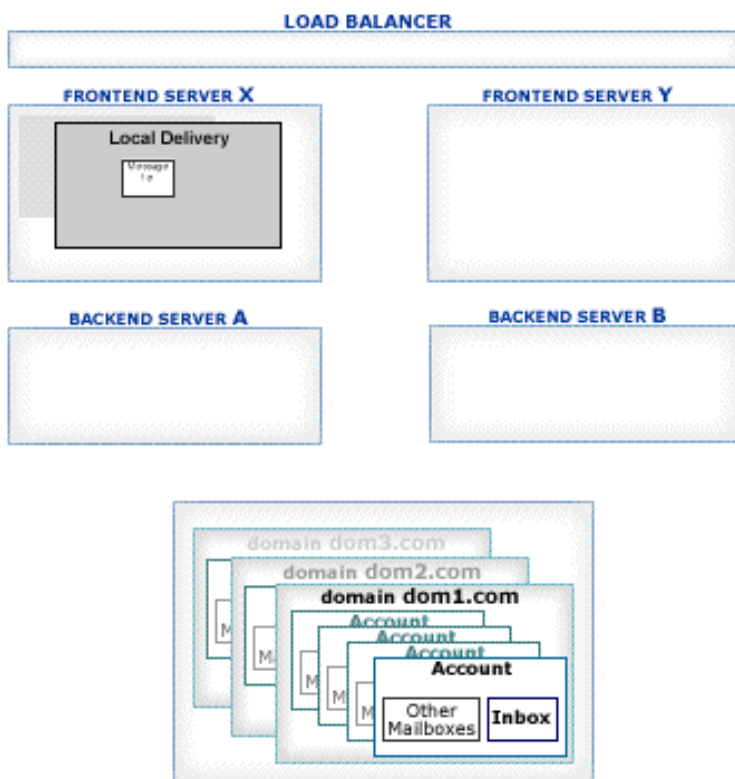
Step 10.



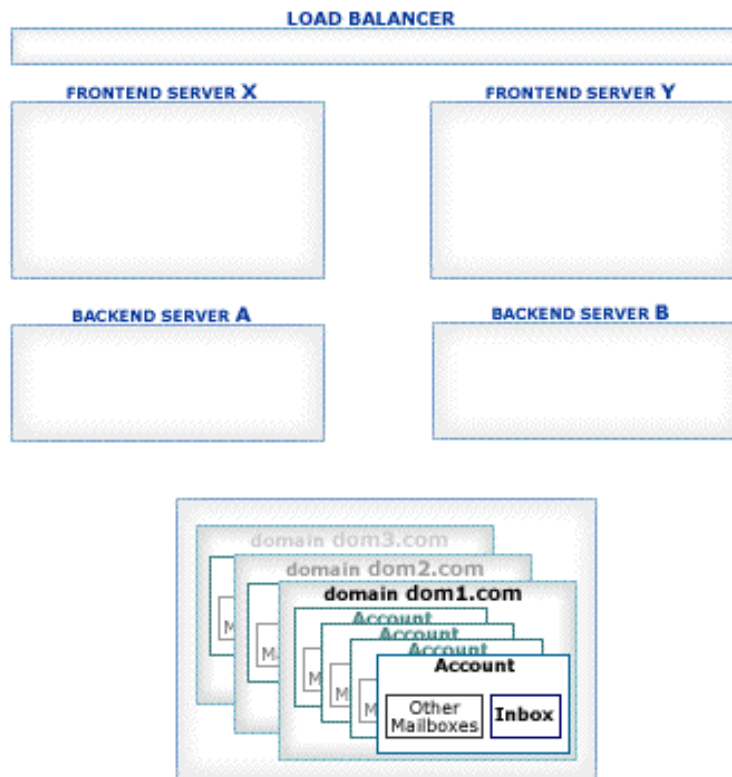
Step 11.



Step 12.



Step 13.



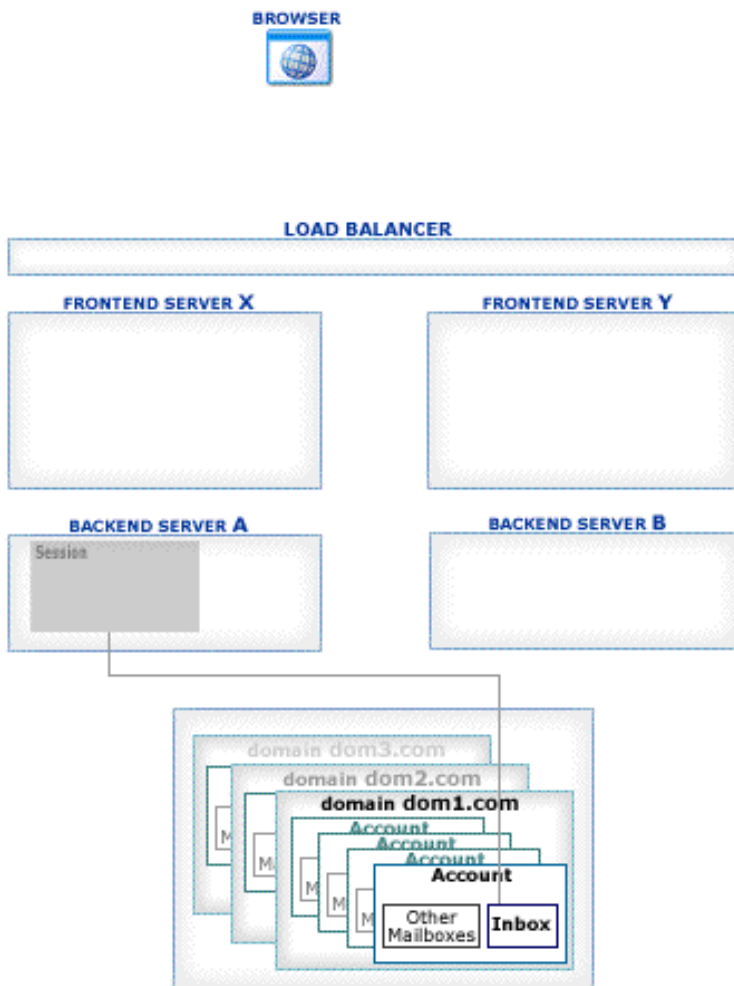
Backend Queues

[WebUser Interface](#) sessions, [Rules](#), [MAPI](#) sessions, and other modules and components can generate E-mail messages on Backend Servers.

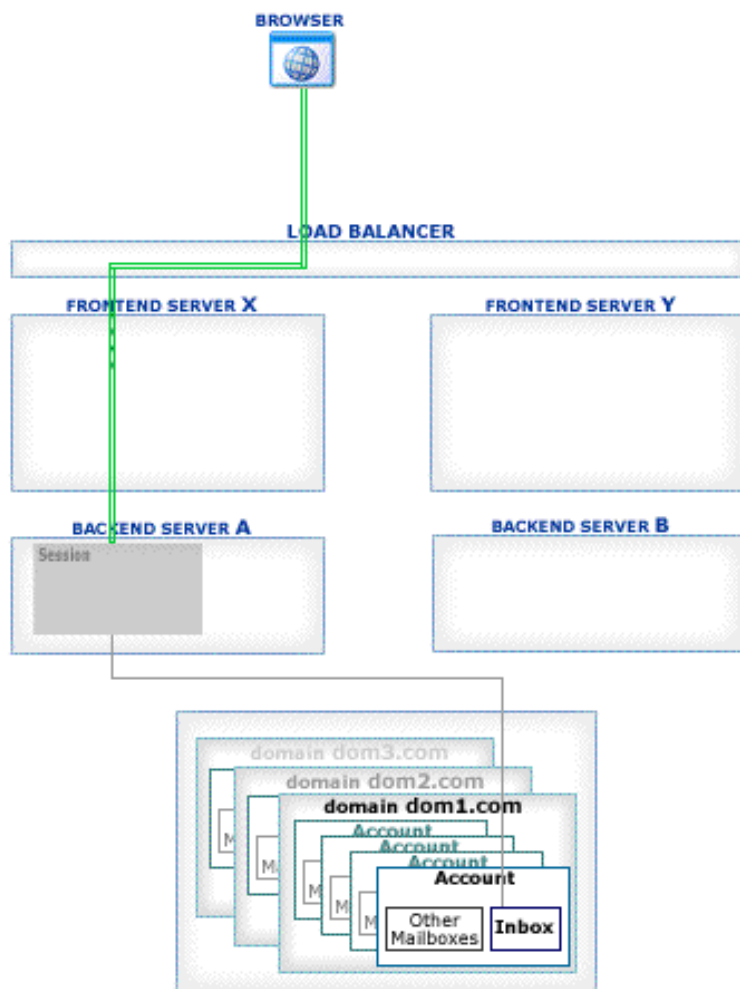
Backend Servers often do not have direct access to the Internet, in this case they cannot deliver generated messages to remote systems. To solve this problem, the Backend Servers can be configured to relay all messages to the Frontend Servers first, using the * symbol as the [SMTP Forwarding Server](#) 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:

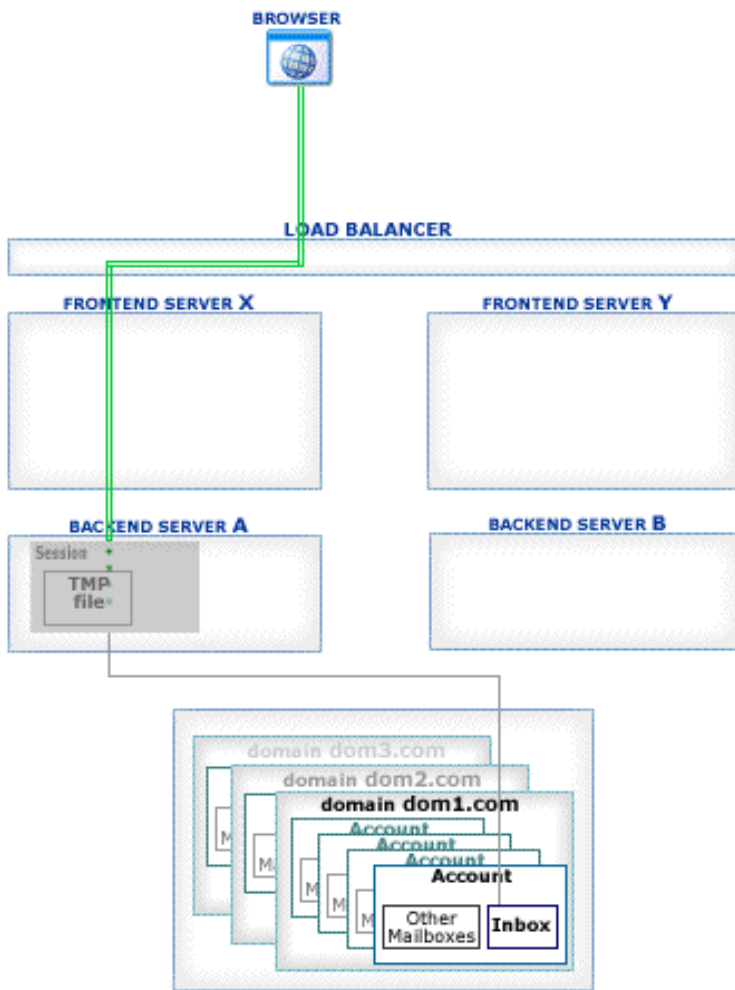
Step 1.



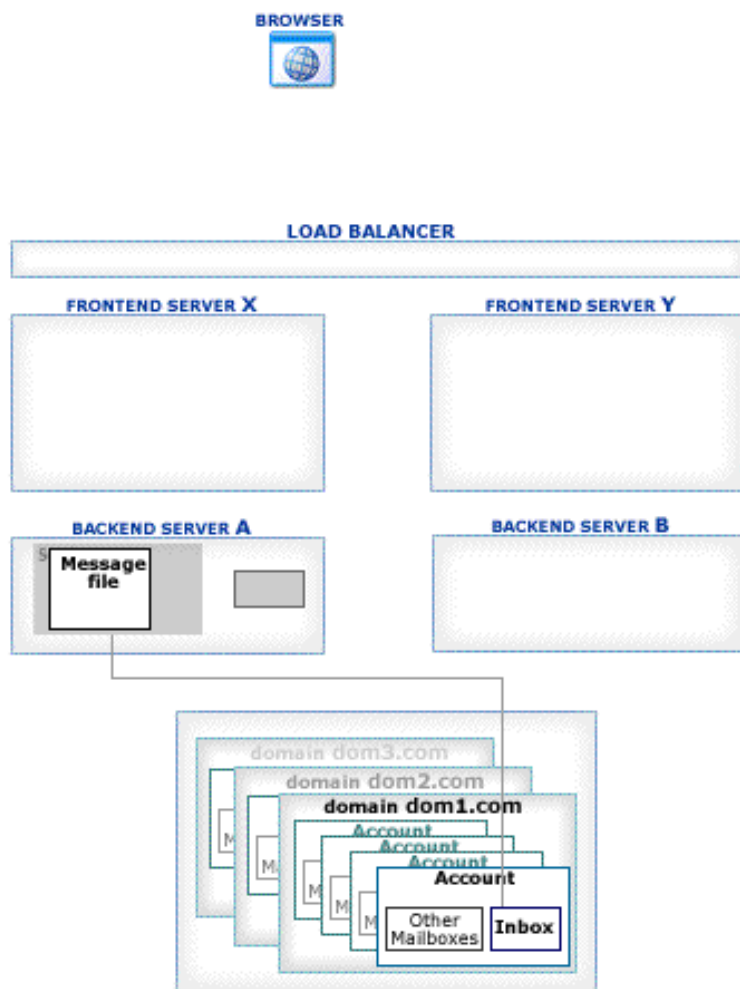
Step 2.



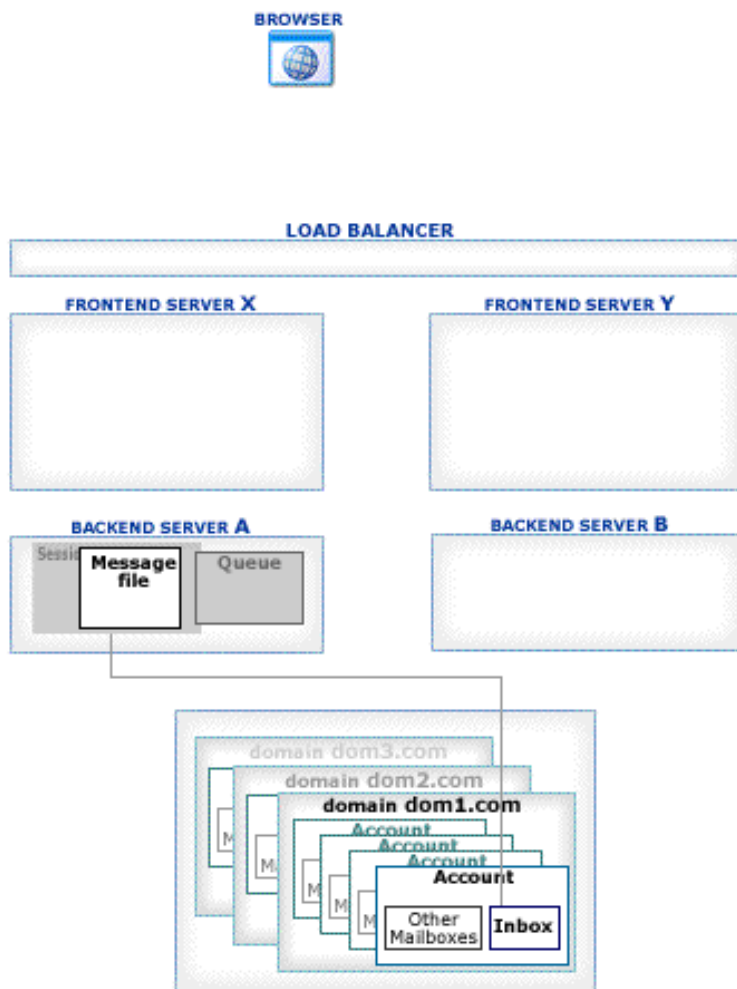
Step 3.



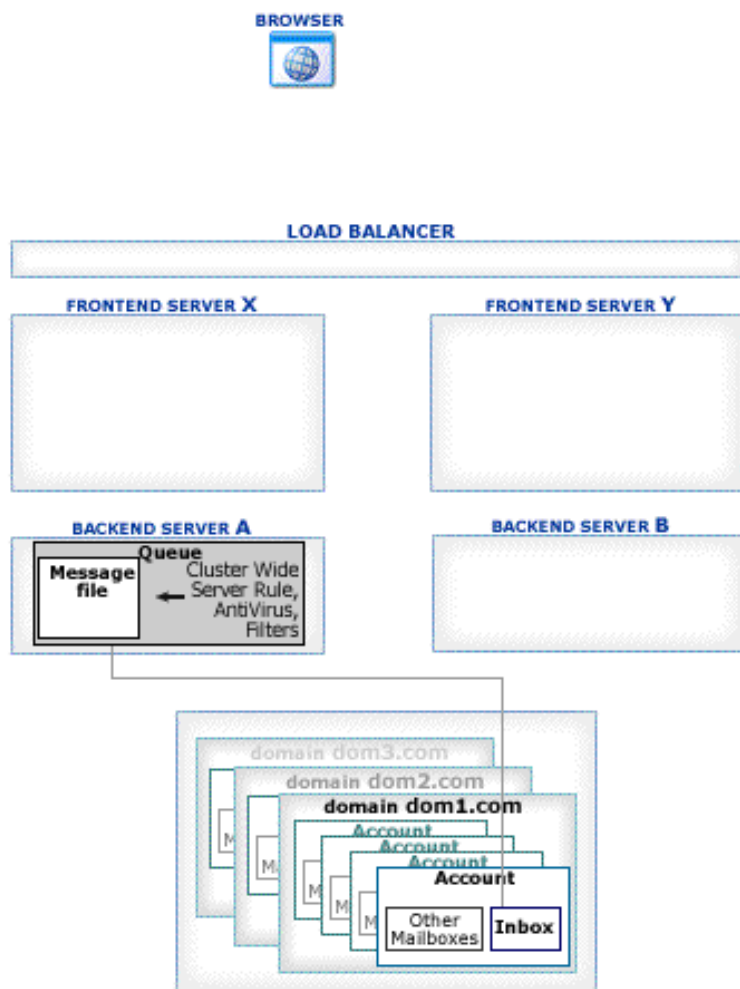
Step 4.



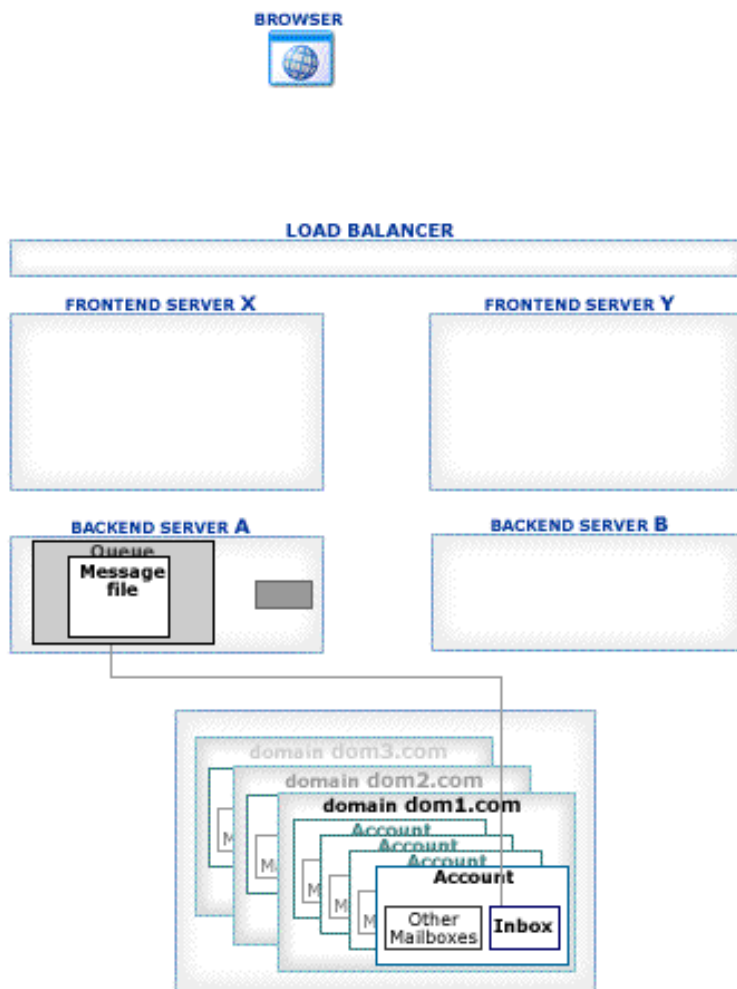
Step 5.



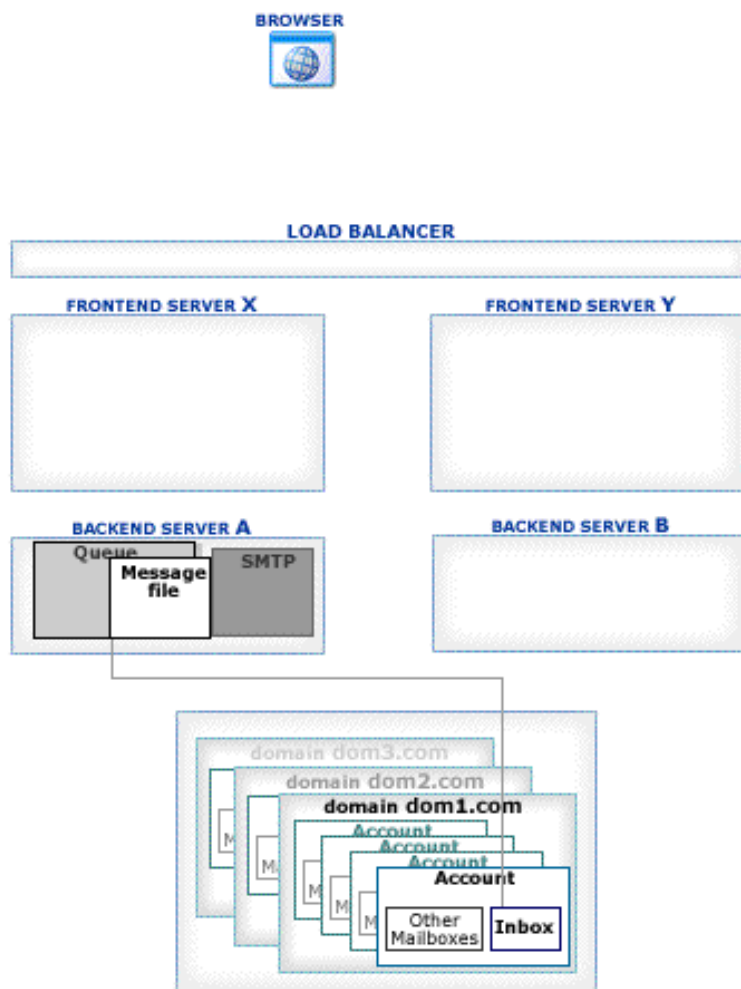
Step 6.



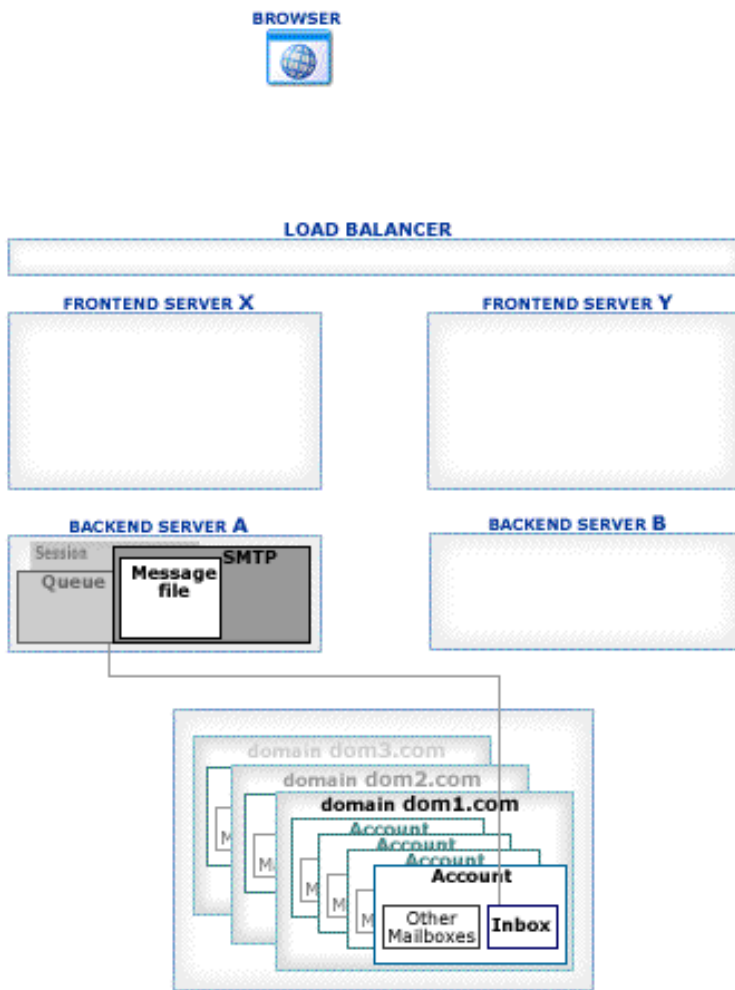
Step 7.



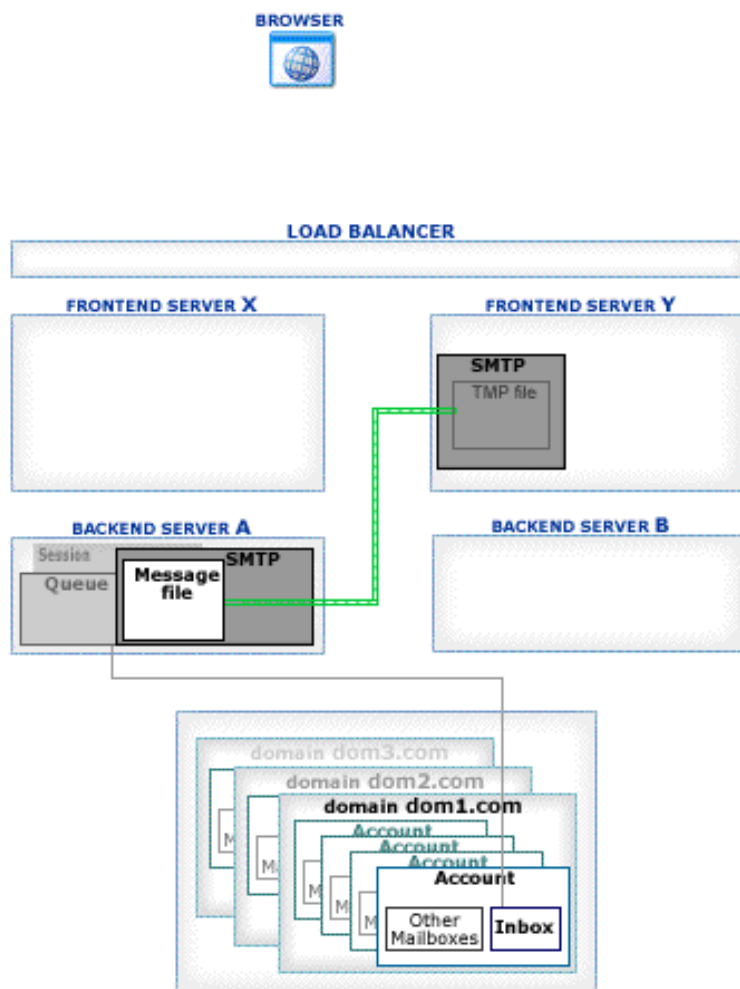
Step 8.



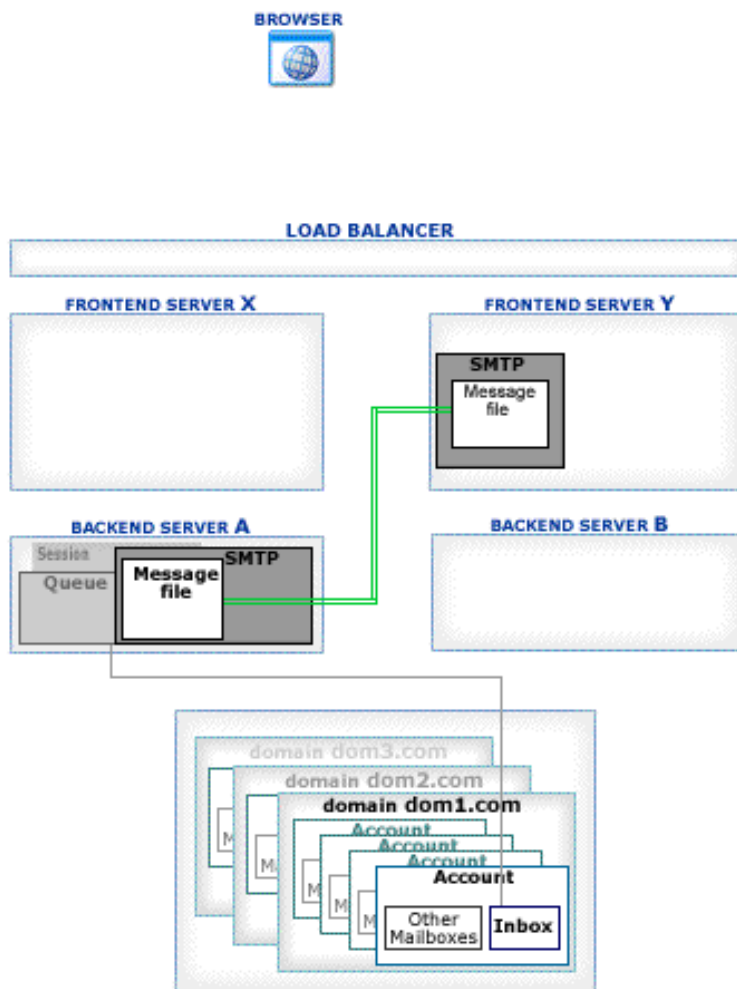
Step 9.



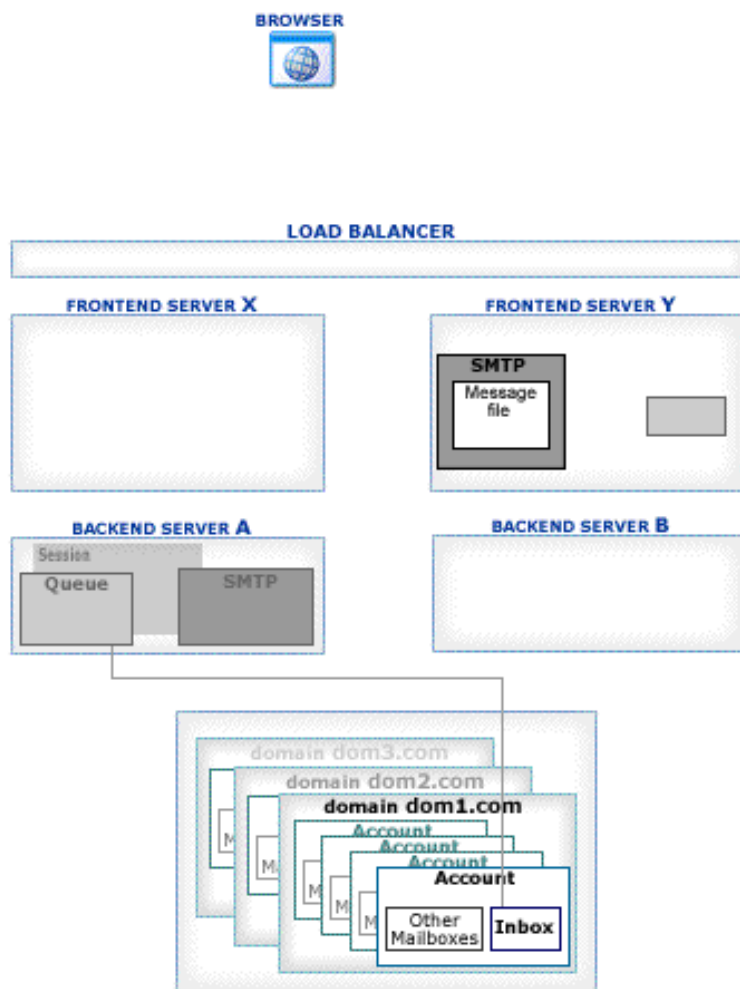
Step 10.



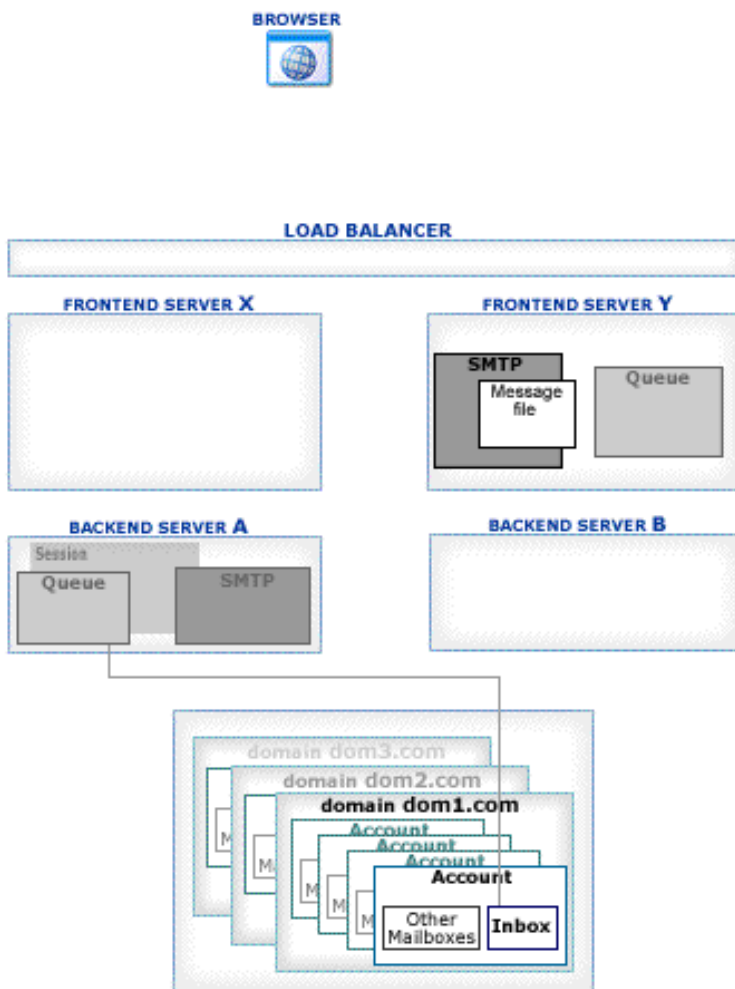
Step 11.



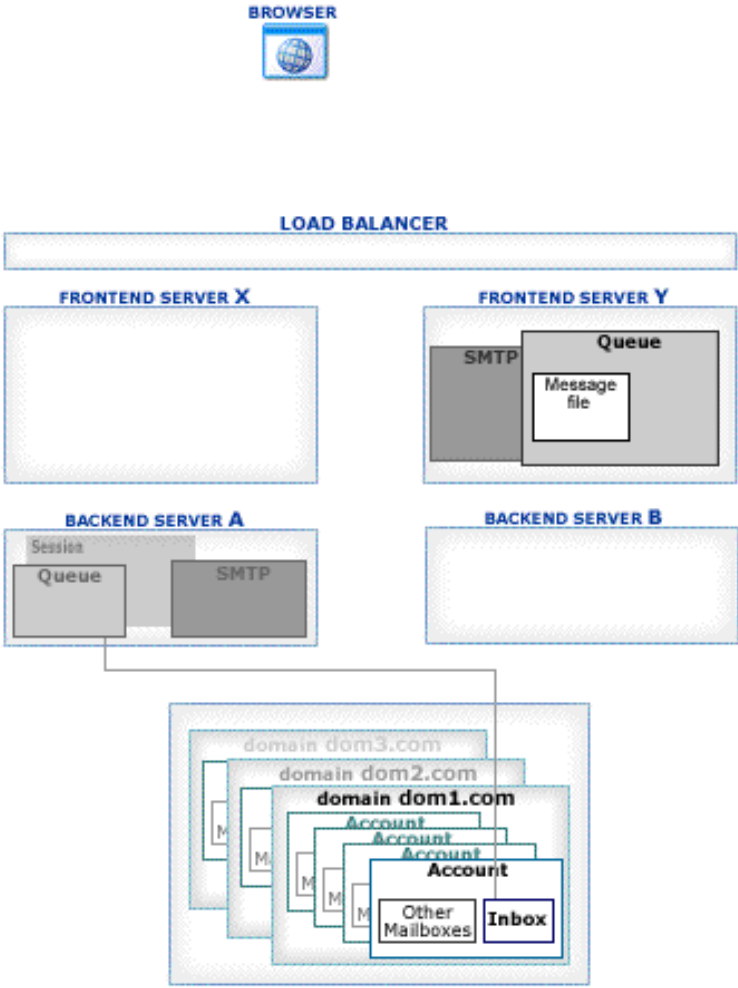
Step 12.



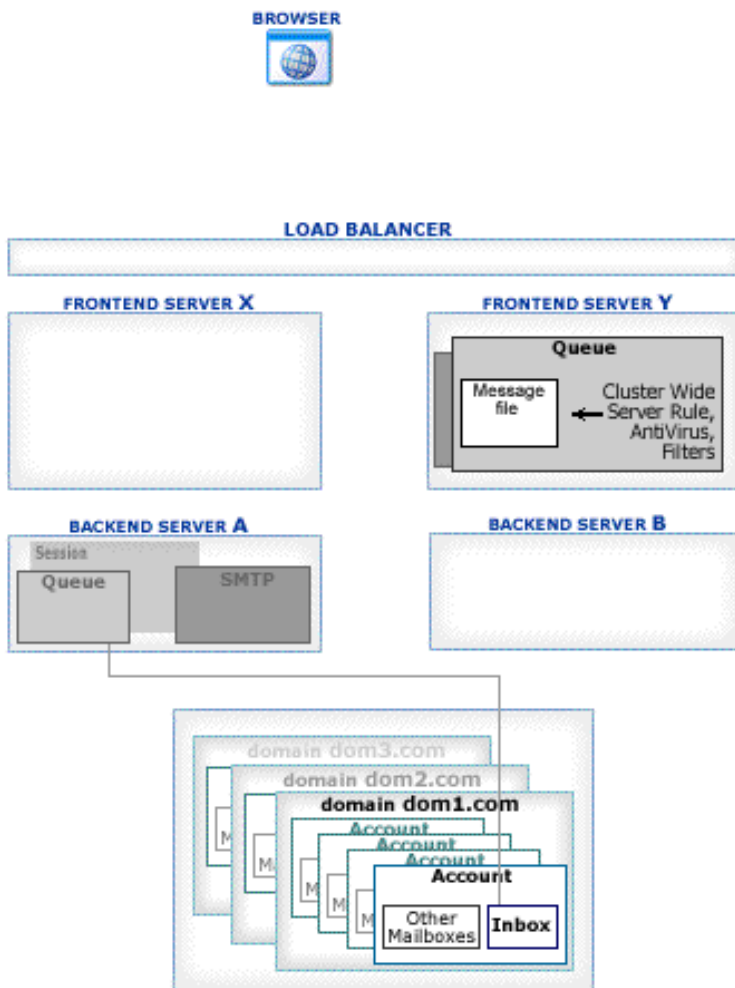
Step 13.



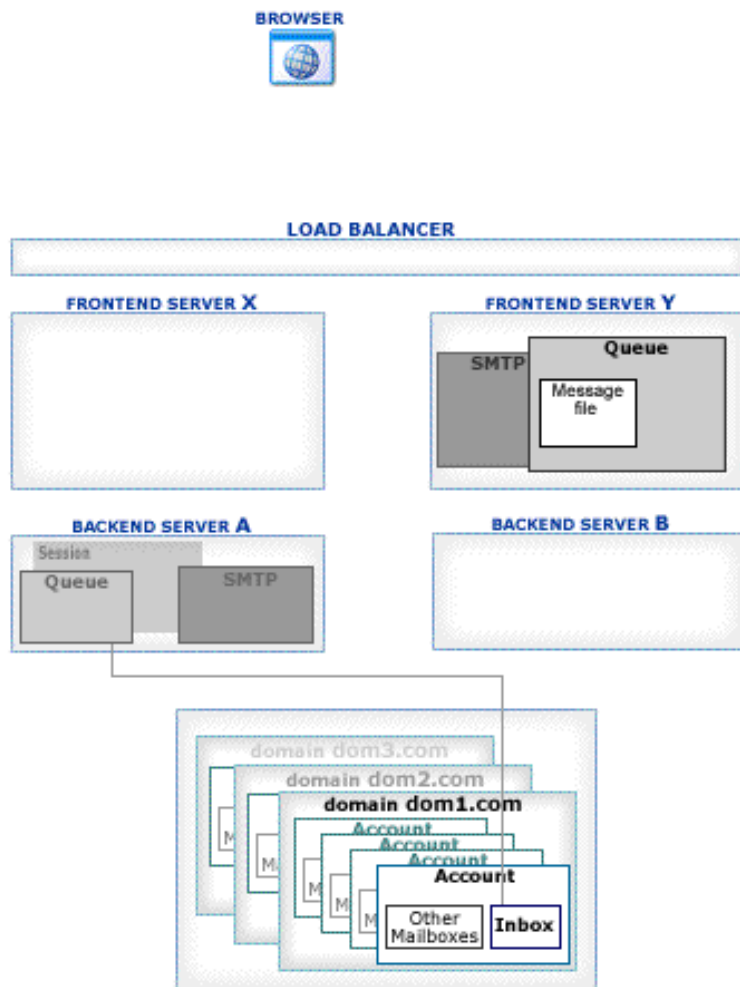
Step 14.



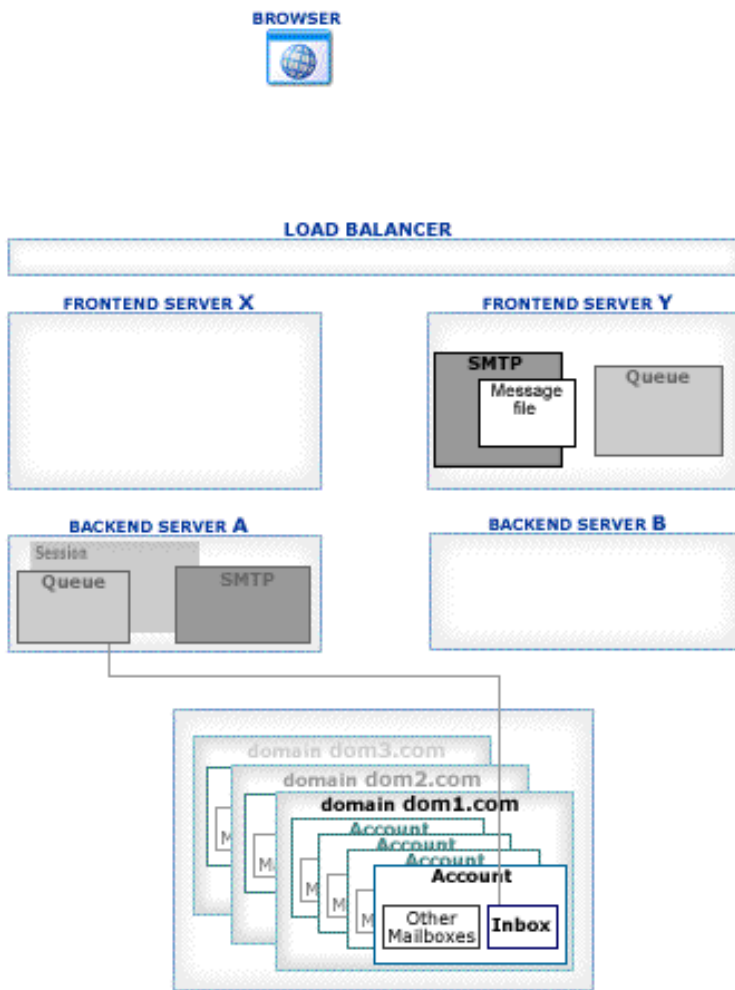
Step 15.



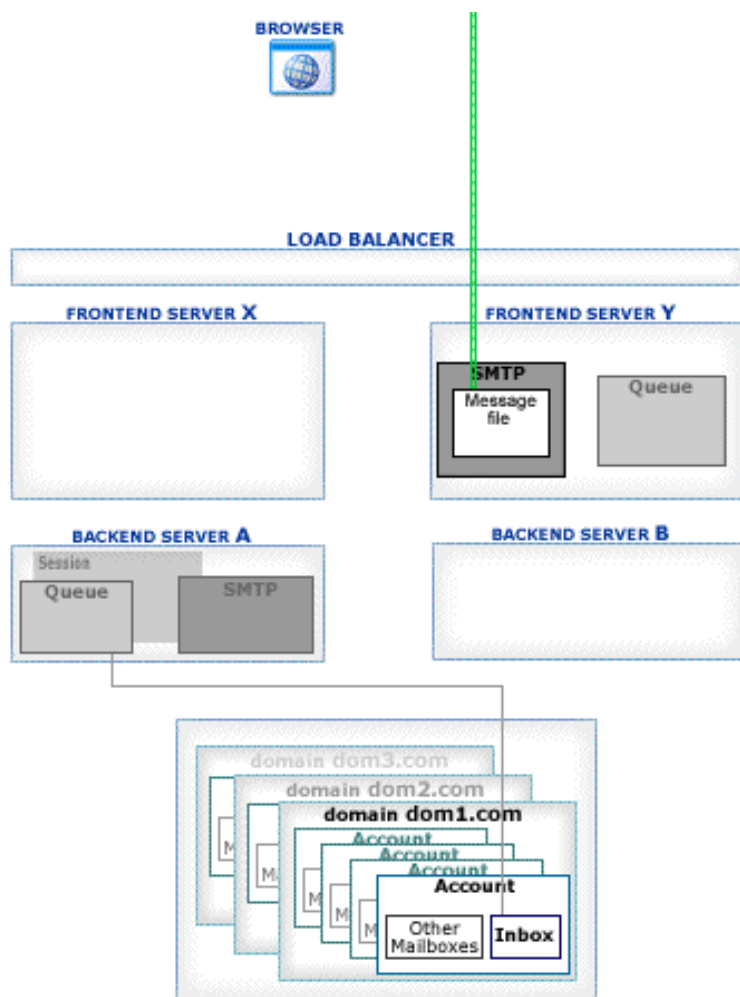
Step 16.



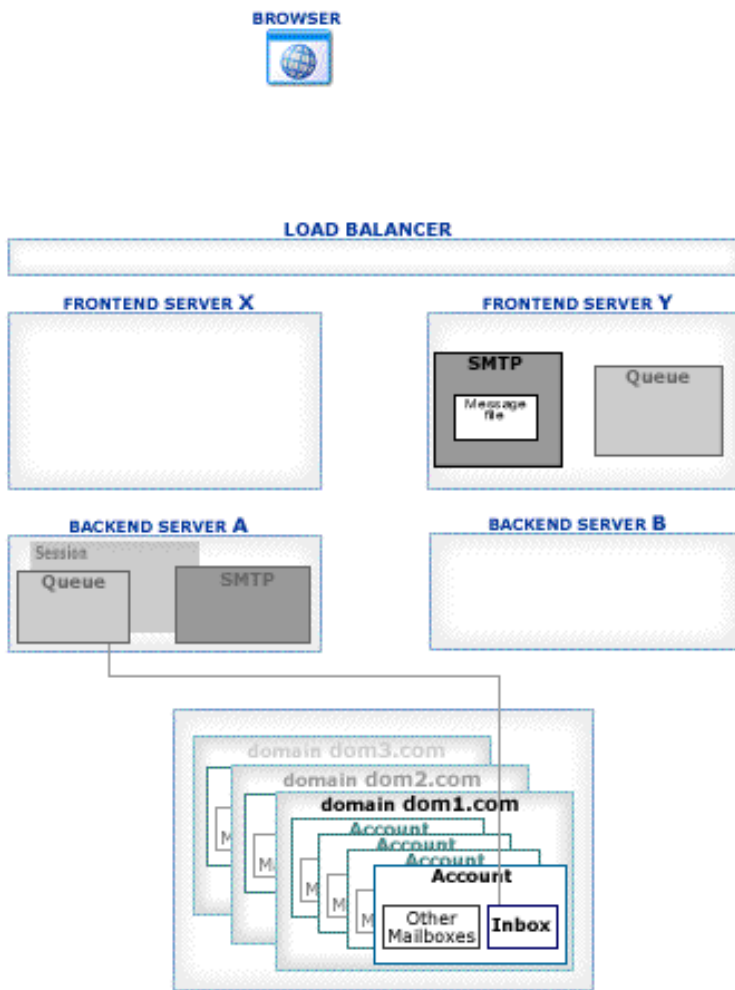
Step 17.



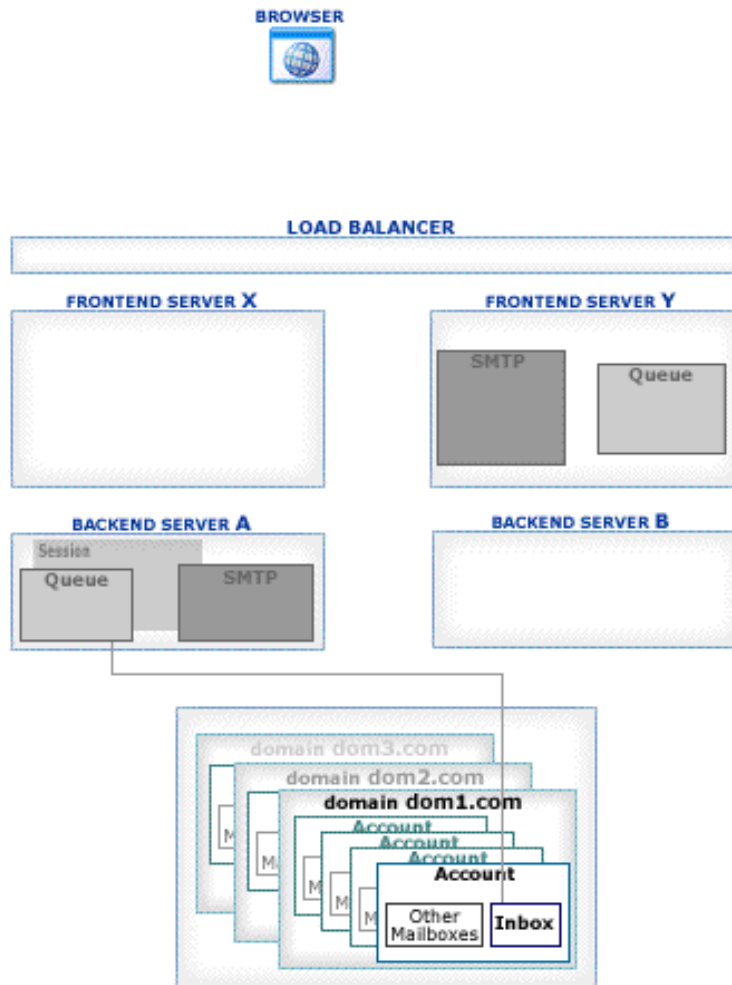
Step 18.



Step 19.



Step 20.



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 this overhead, the Remote Queue processing method can be used.

Remote Queue Processing

Most of the Queue processing takes place on the Frontend Servers. Frontend Servers accept incoming SMTP 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 messages can be generated directly on the Backend Servers. They include messages:

- created with Account-level Rules;
- composed within WebUser sessions;
- retrieved using the RPOP module;
- submitted using the MAPI connector and XTND XMIT POP3 mechanism;
- 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:	Locally <input type="button" value="v"/>

Submit Messages

This setting specifies how the Queue should be processed by this Cluster member.

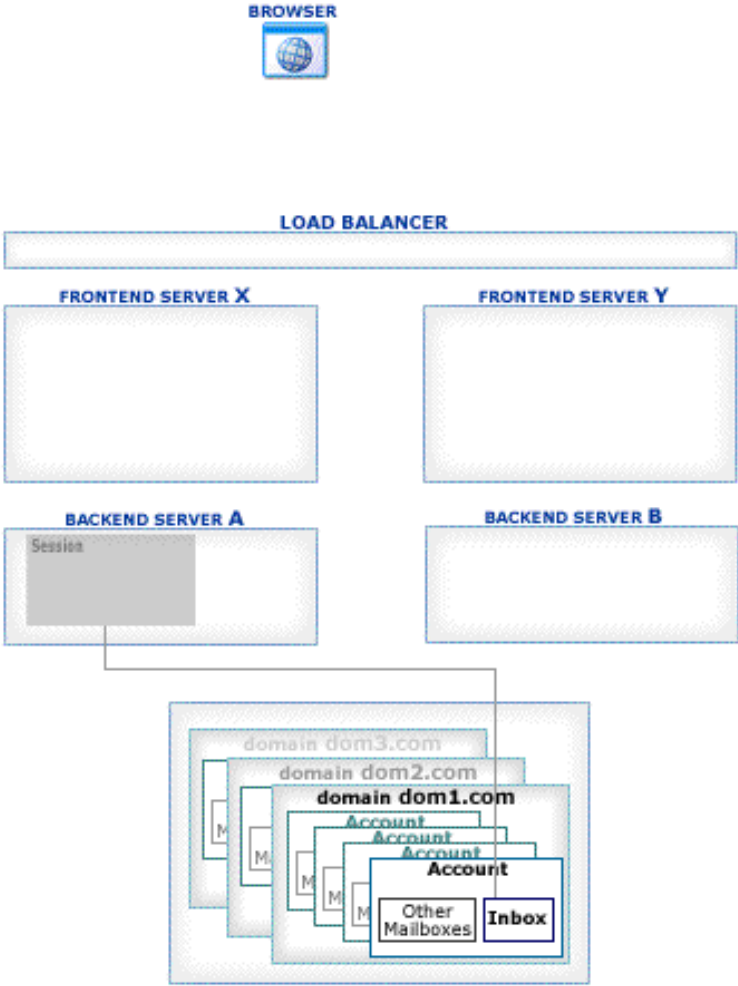
If the Locally option is selected, messages will be processed in the "regular" way: when a message is

composed in a temporary file, it is submitted into the Queue maintained by this Server.

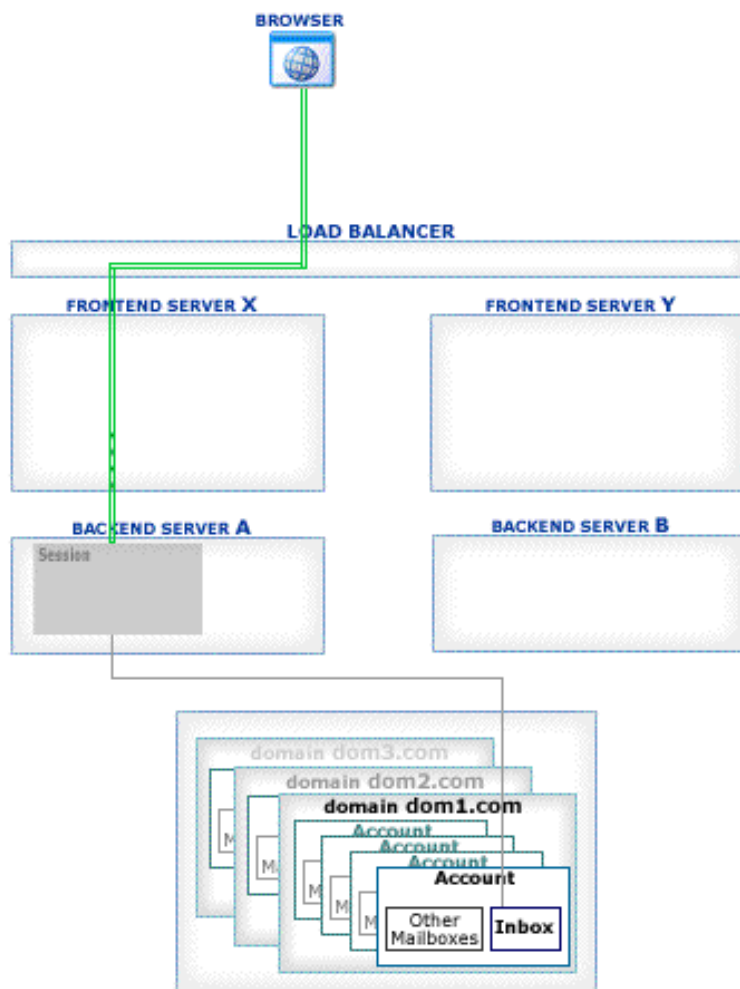
If the Locally for Others option is selected, messages will be processed in the "regular" way, too. In addition, this Server will accept messages generated on other Cluster members and submit them into its own Queue - for processing and delivery. The Dynamic Cluster Controller collects and distributes information about all running Cluster members that have this option selected.

If the Remotely option is selected, this Cluster member will try to connect to any other Cluster member that has the "Locally for Others" option selected. The temporary 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:

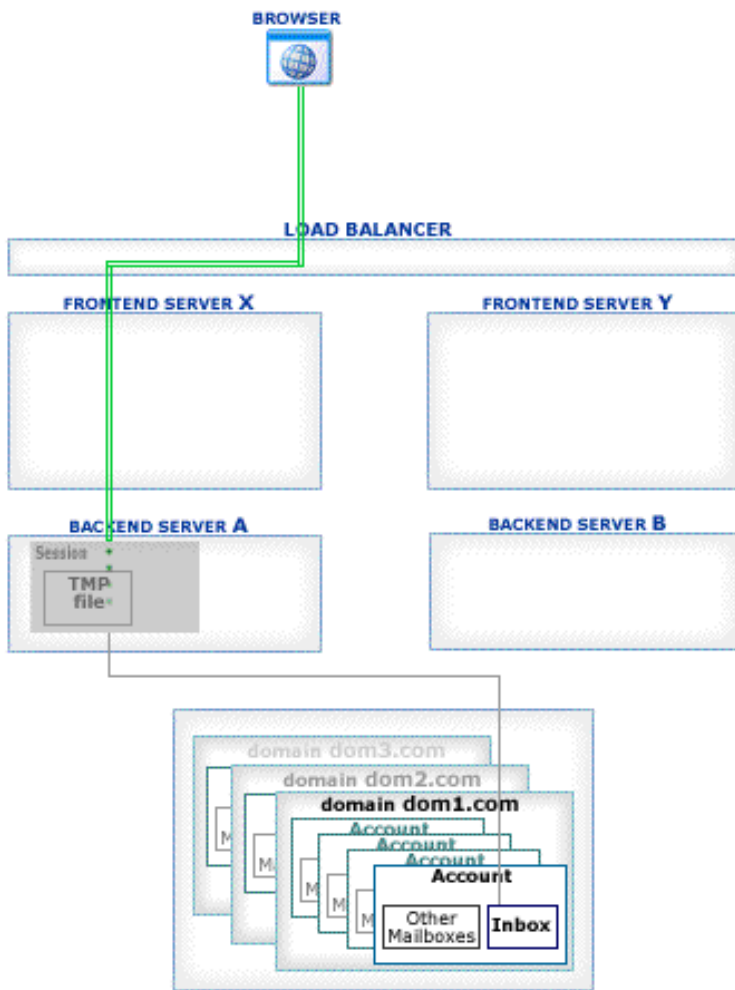
Step 1.



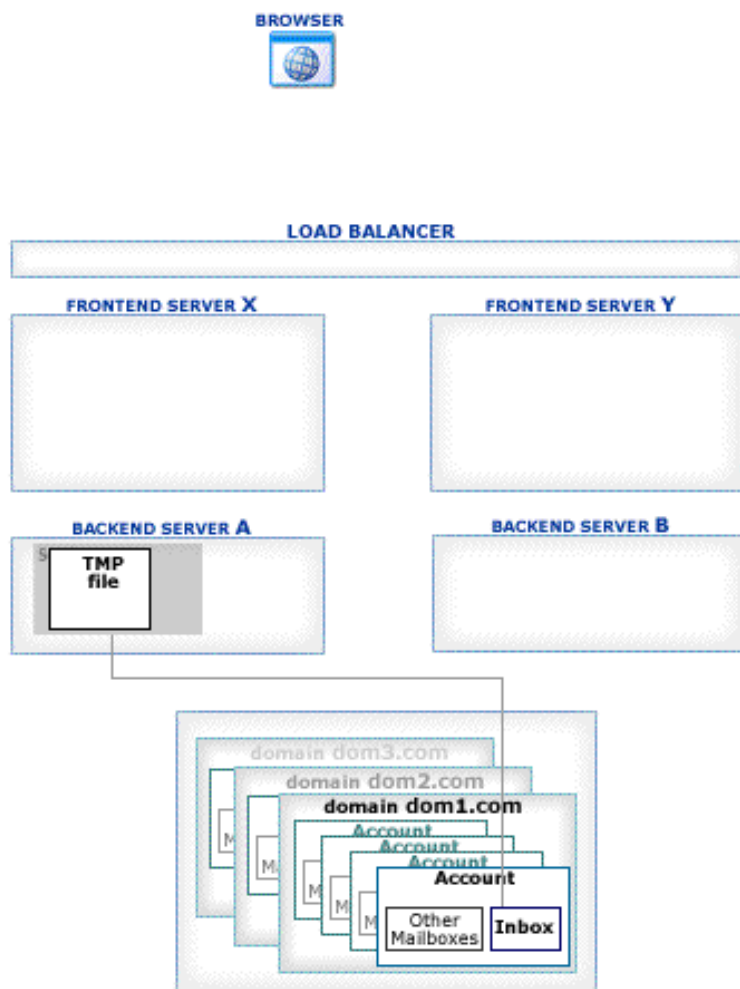
Step 2.



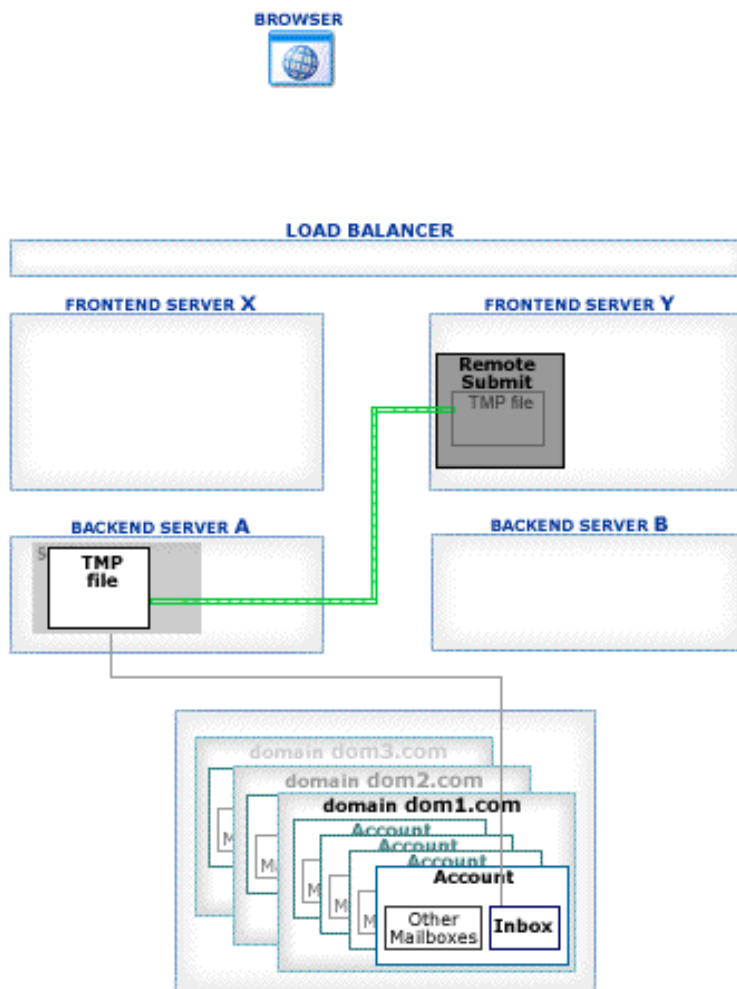
Step 3.



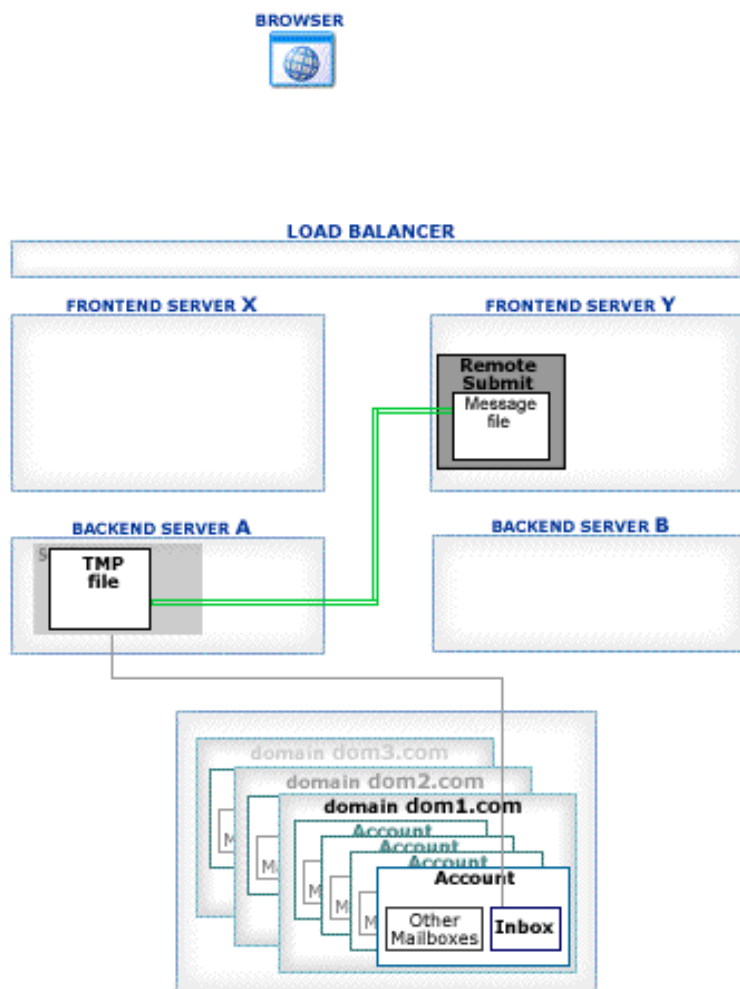
Step 4.



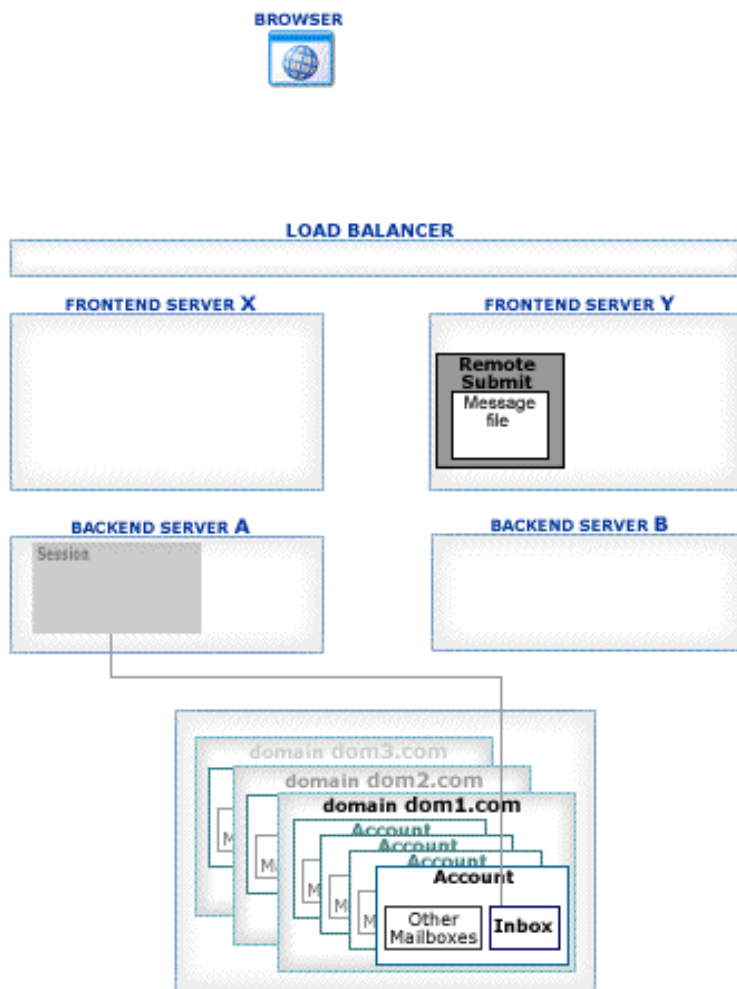
Step 5.



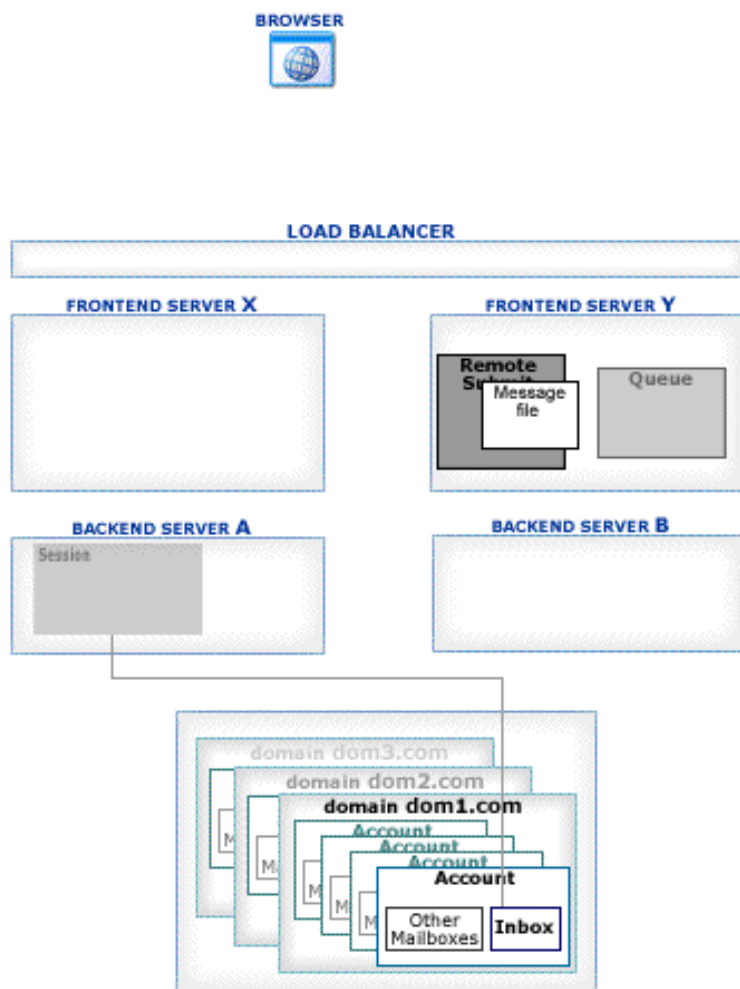
Step 6.



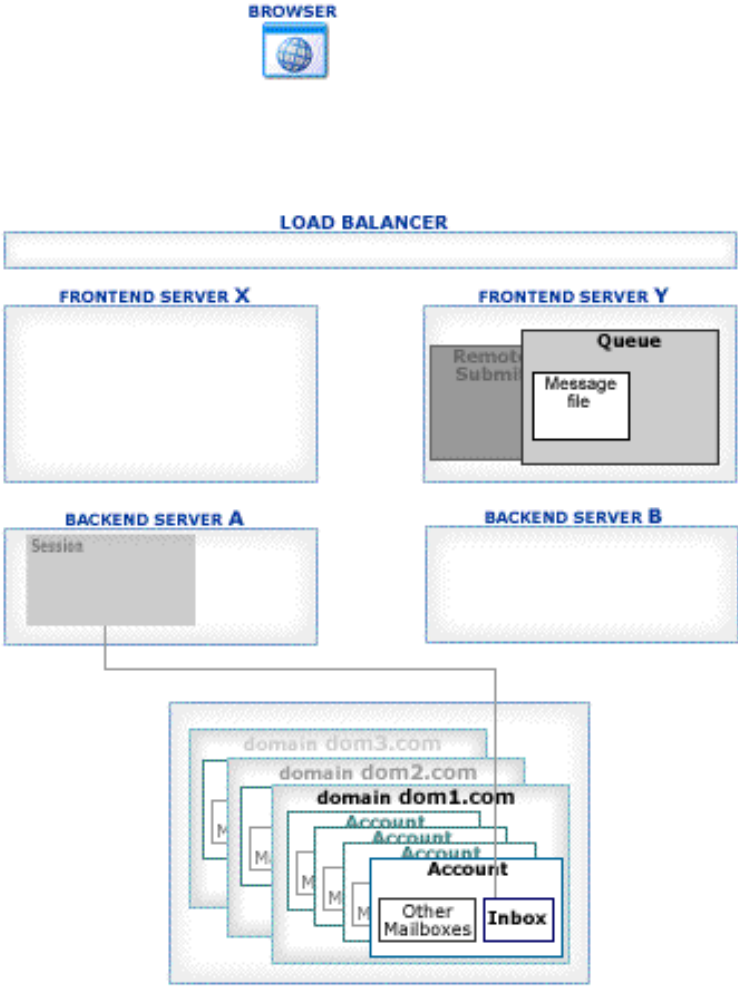
Step 7.



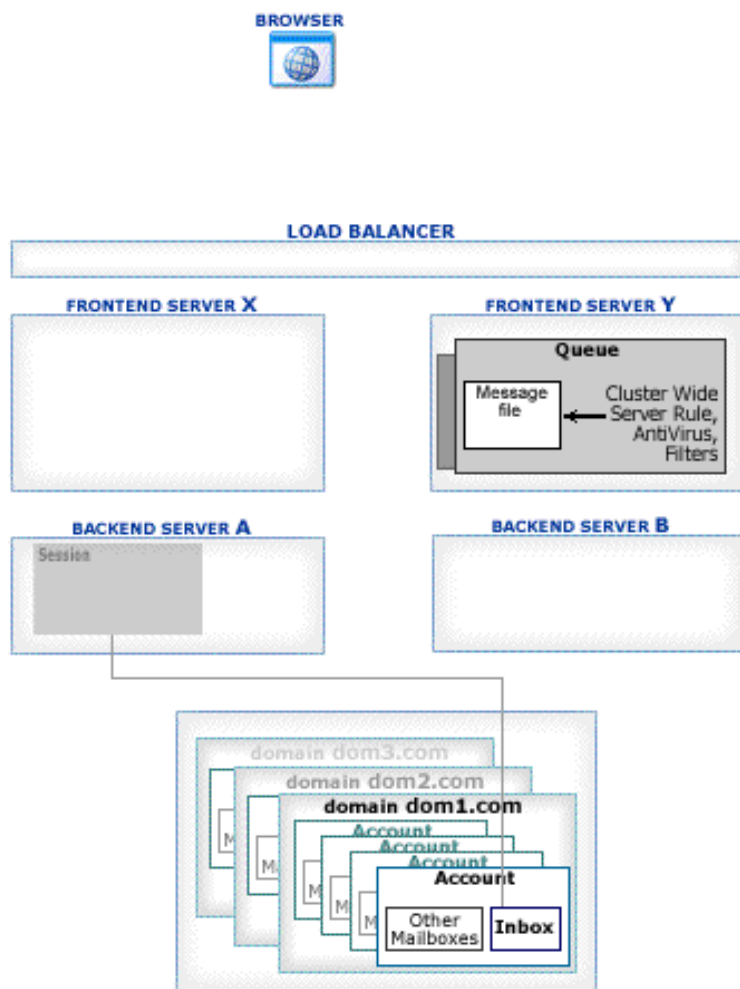
Step 8.



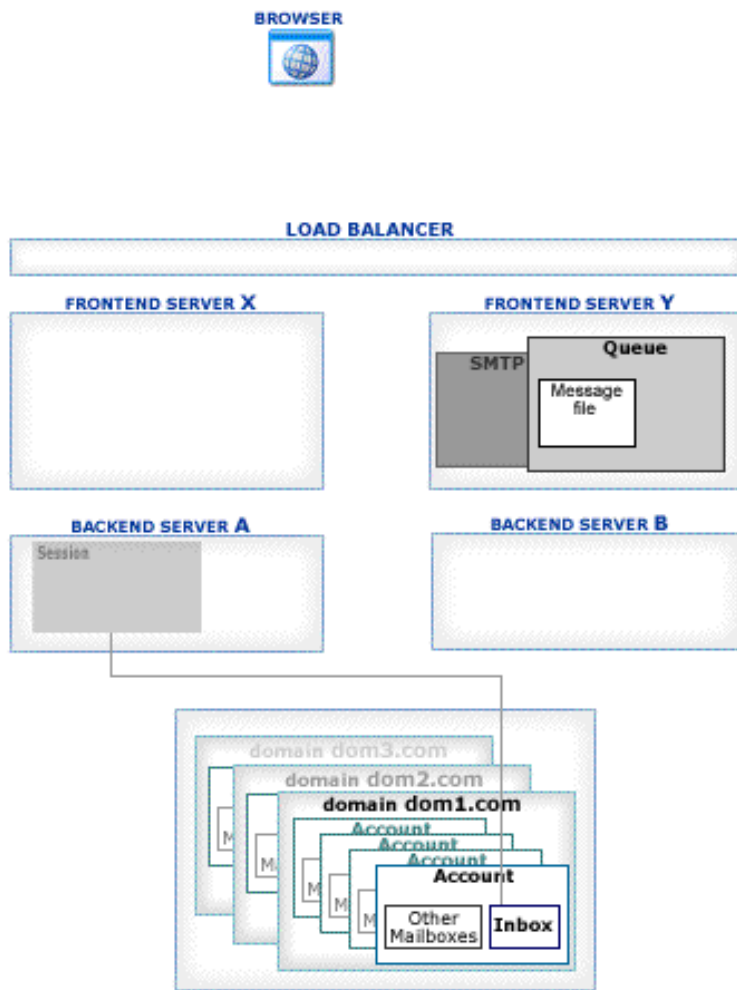
Step 9.



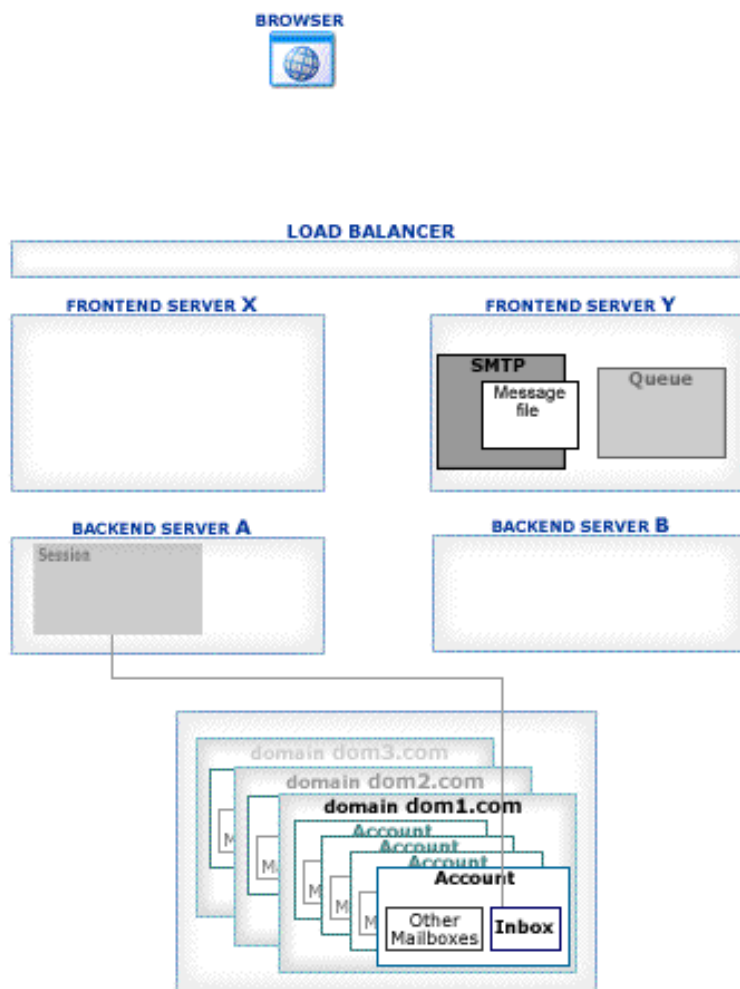
Step 10.



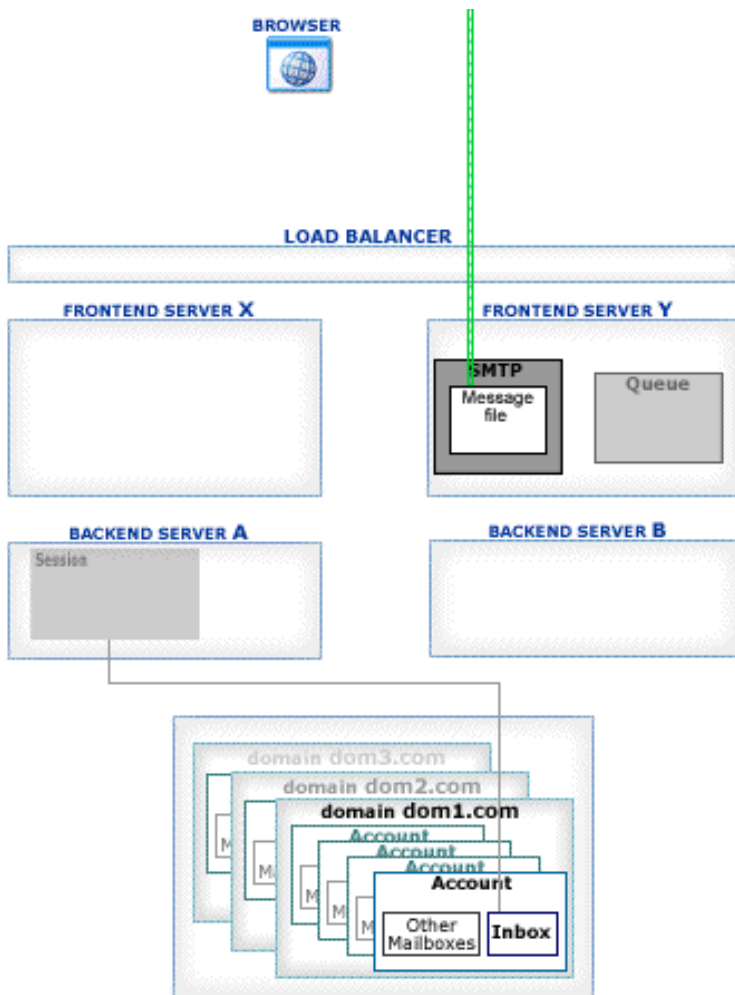
Step 11.



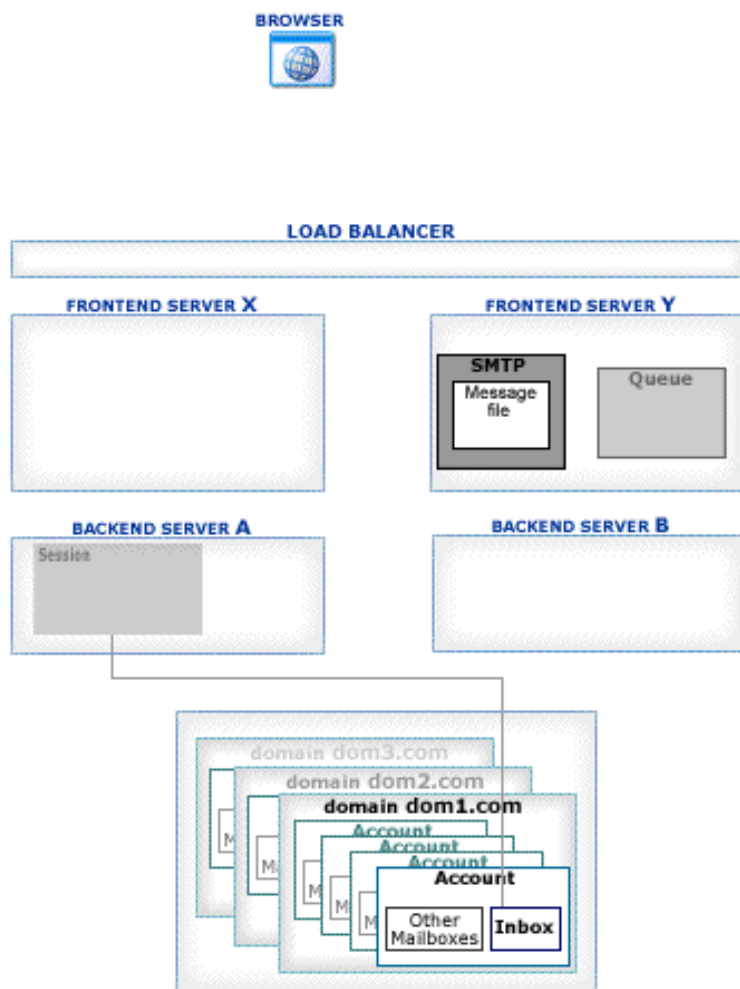
Step 12.



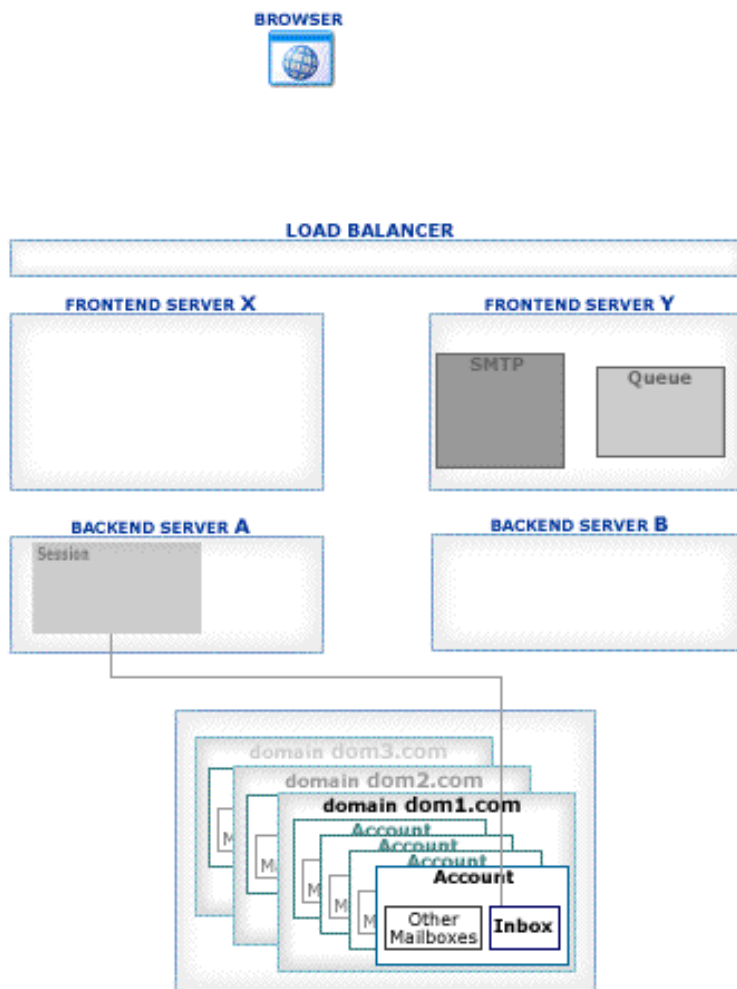
Step 13.



Step 14.



Step 15.





Cluster Signals

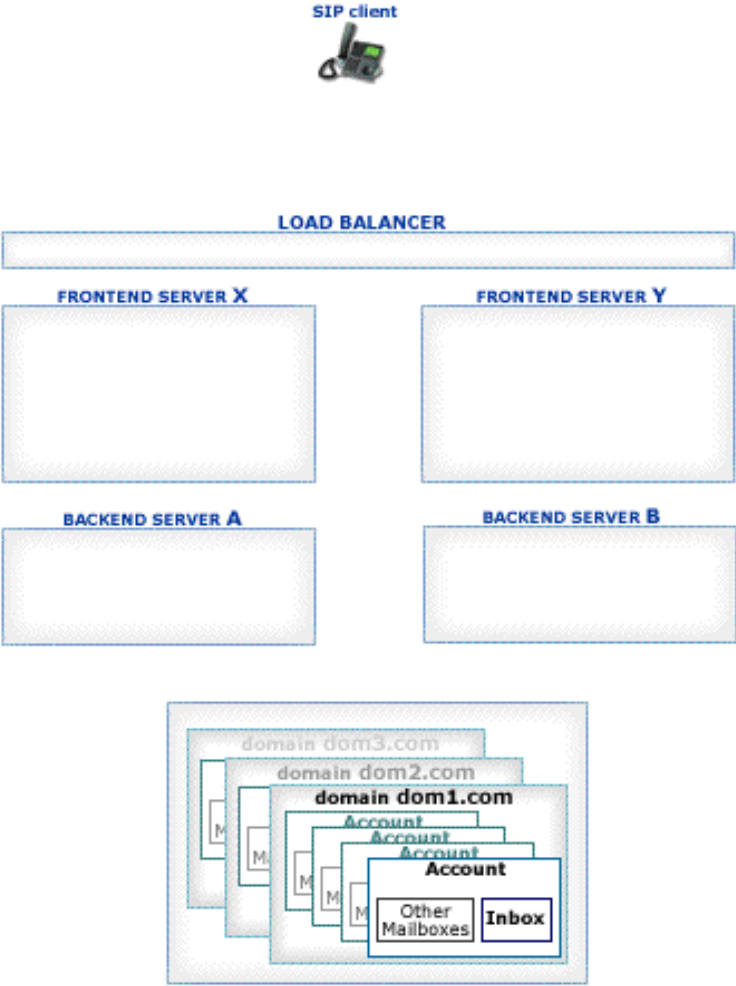
This section explains how [Real Time Communication](#) operations work in the CommuniGate Pro Cluster environment.

SIP

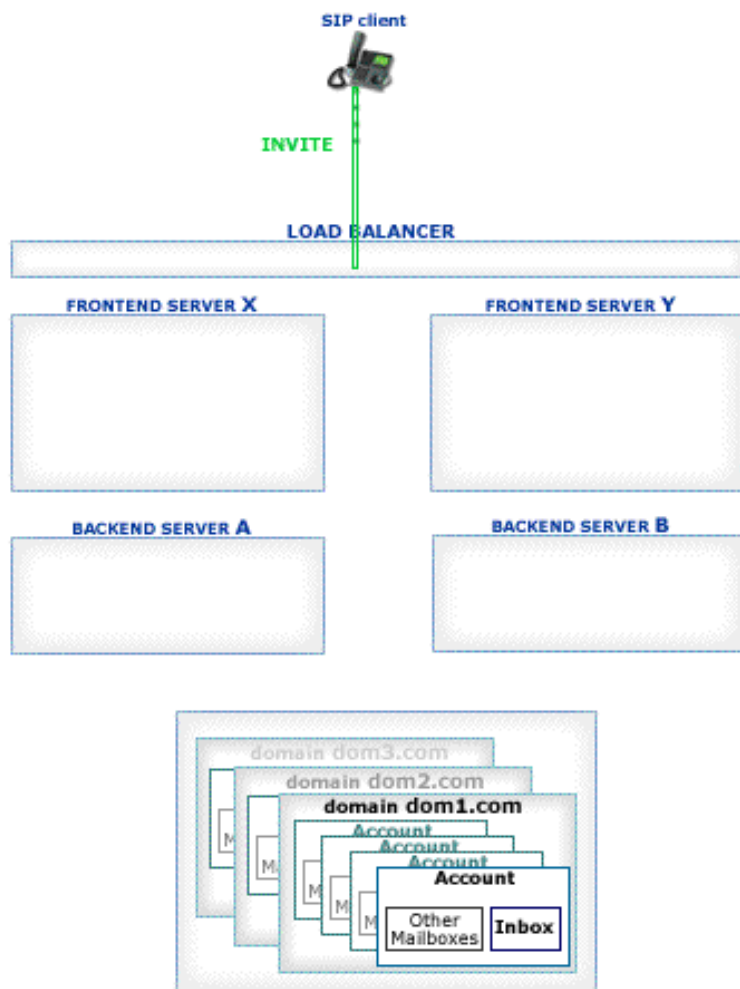
The CommuniGate Pro Cluster maintains the information about all its Servers with the SIP Farm option switched on. Incoming UDP packets and TCP connections are distributed to those Servers using regular simple Load Balancers.

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 Farm Server:

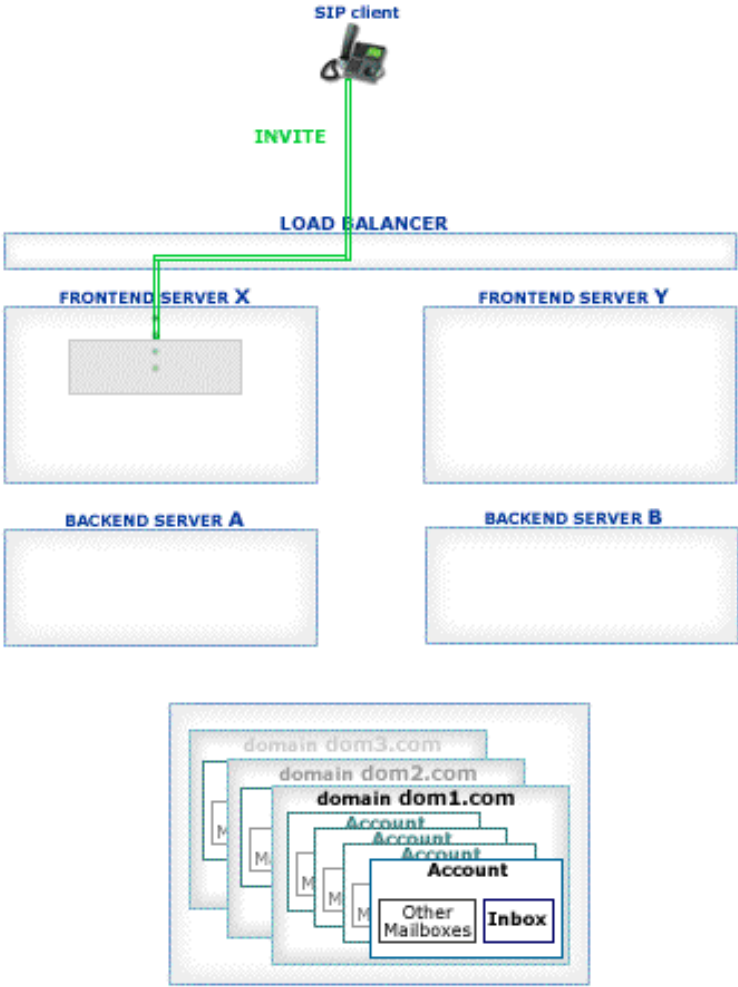
Step 1.



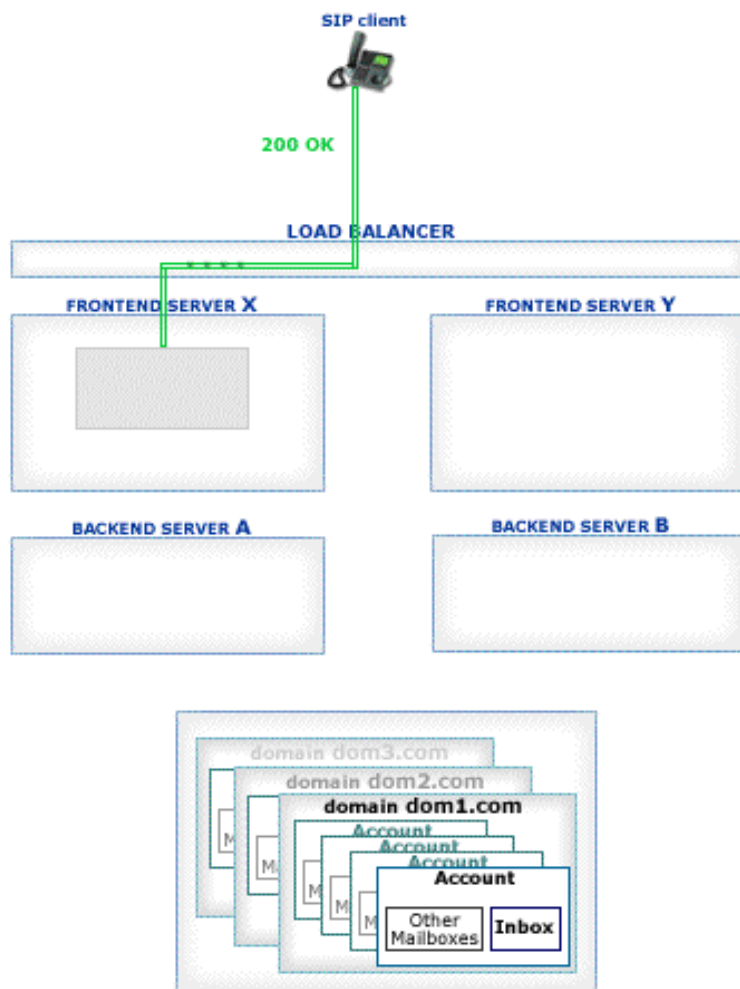
Step 2.



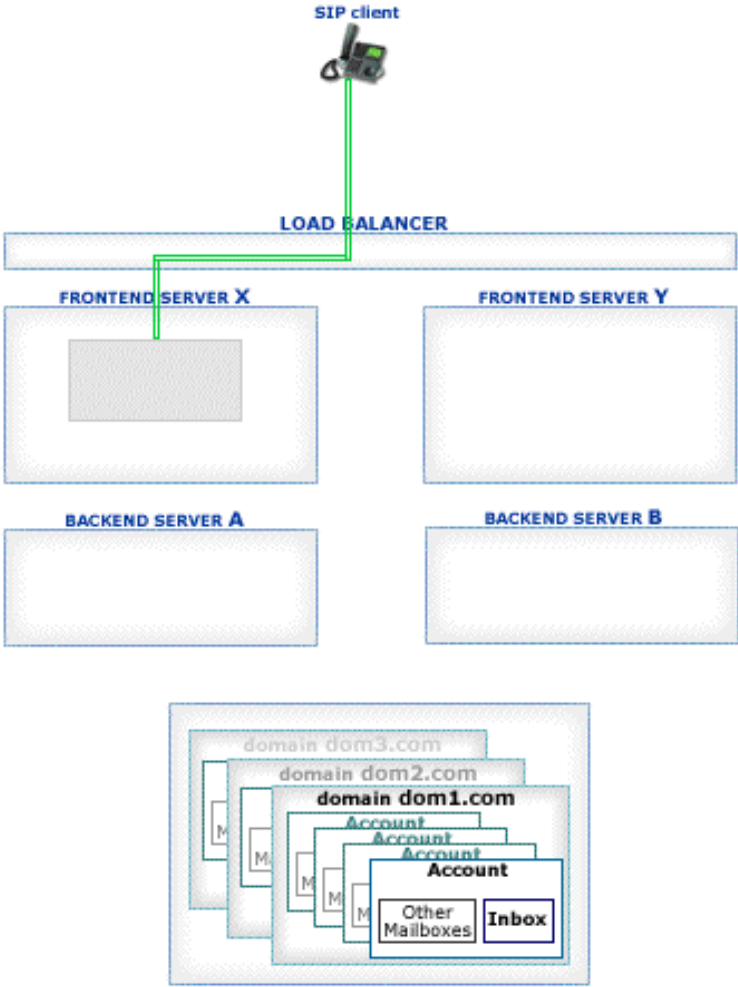
Step 3.



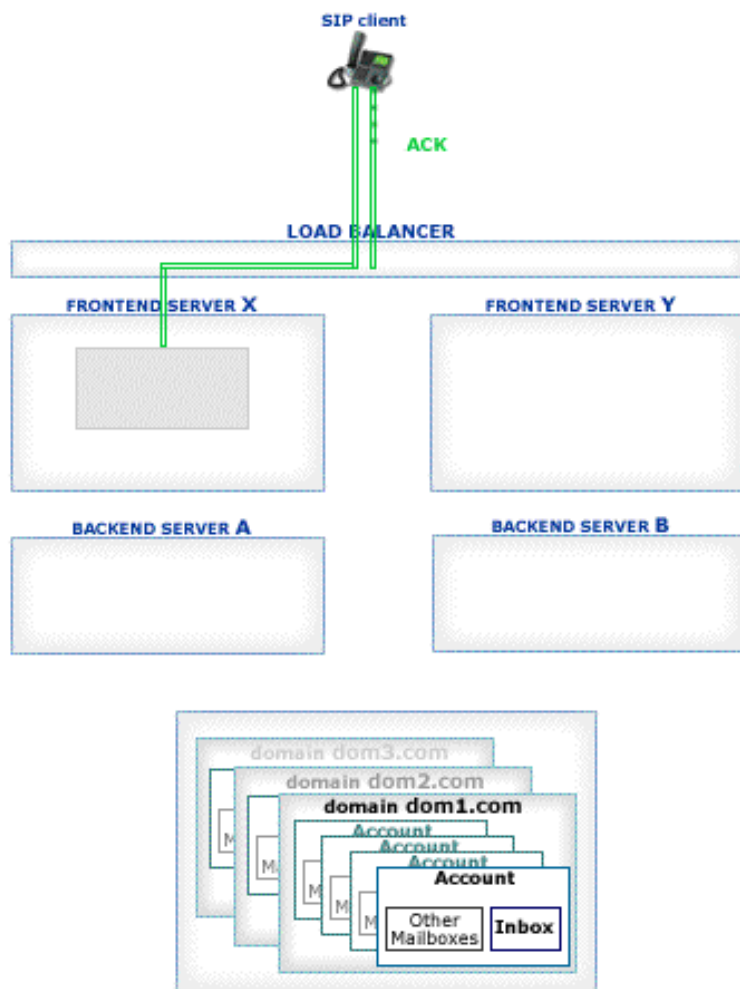
Step 4.



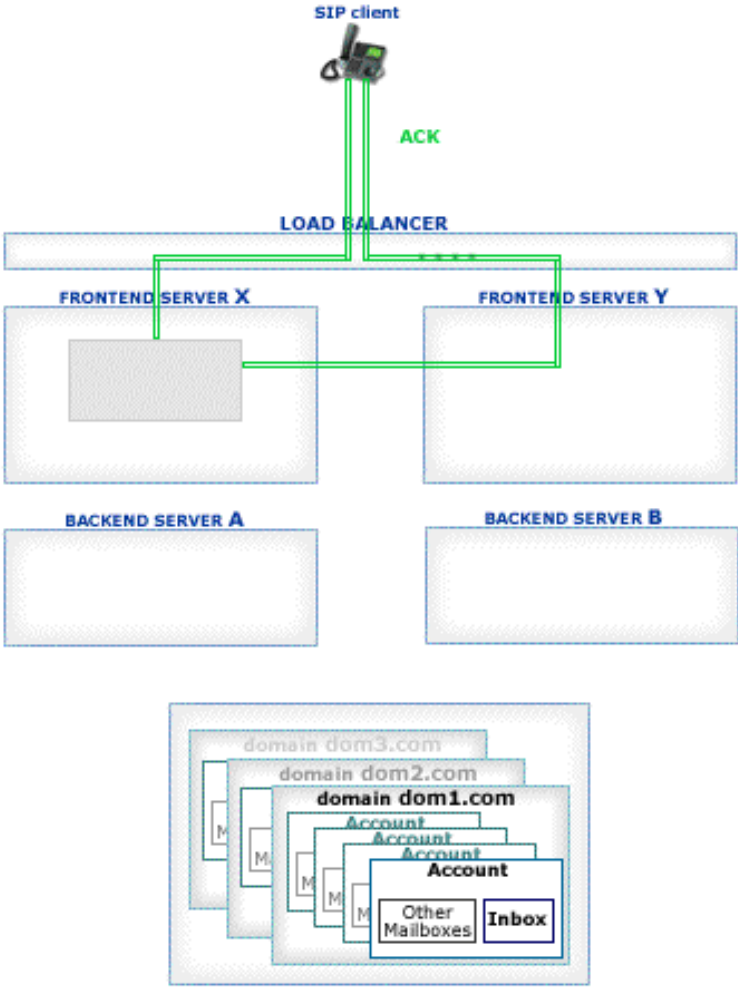
Step 5.



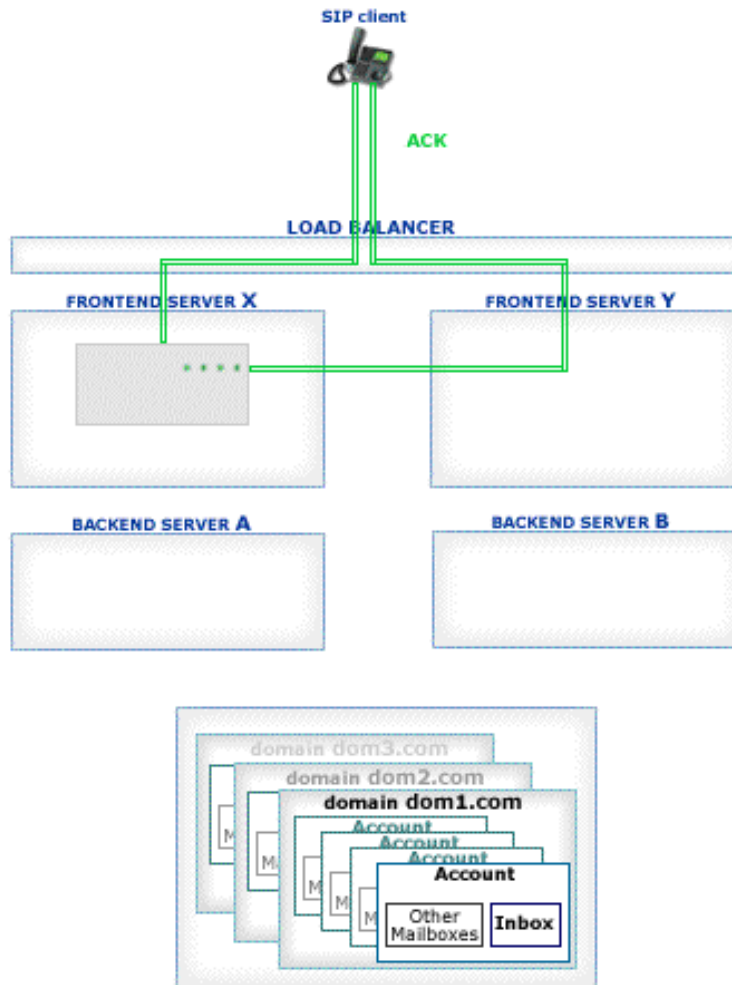
Step 6.



Step 7.



Step 8.



Packets not directed to a particular Farm Server are distributed to all Farm Members based on the CommuniGate Pro Cluster algorithms and the currently available set of the active Farm Members.

The Farm Members may need to retrieve certain Account information (registration, preferences, etc.). If a Farm

Member cannot open the Account (because the Member is a Frontend Server or because the Account is locked on a different Backend Server), the Member uses Inter-Cluster CLI to retrieve the required information from the proper Backend Server.



Cluster Access

This section explains how [Account Access](#) operations work in the CommuniGate Pro Cluster environment.

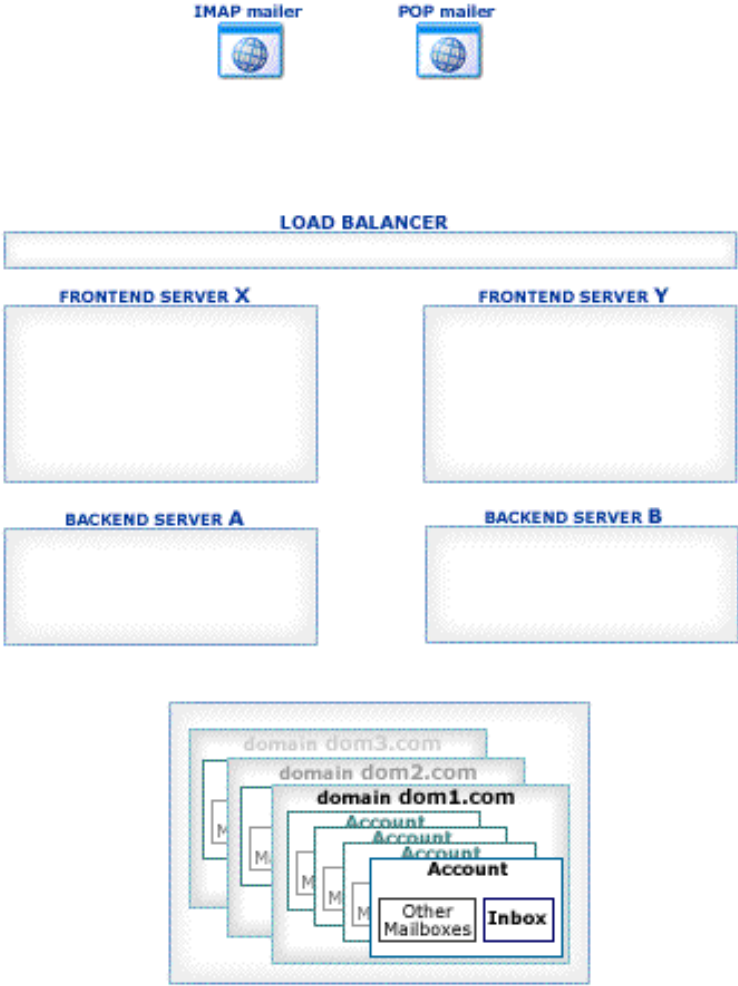
The CommuniGate Pro Cluster architecture allows a load balancer to direct any connection to any working Server, eliminating a need for complex and unreliable "high-level" load balancer solutions. Inexpensive Layer 4 Switches can be used to handle the traffic.

POP, IMAP, MAPI, ACAP Interfaces

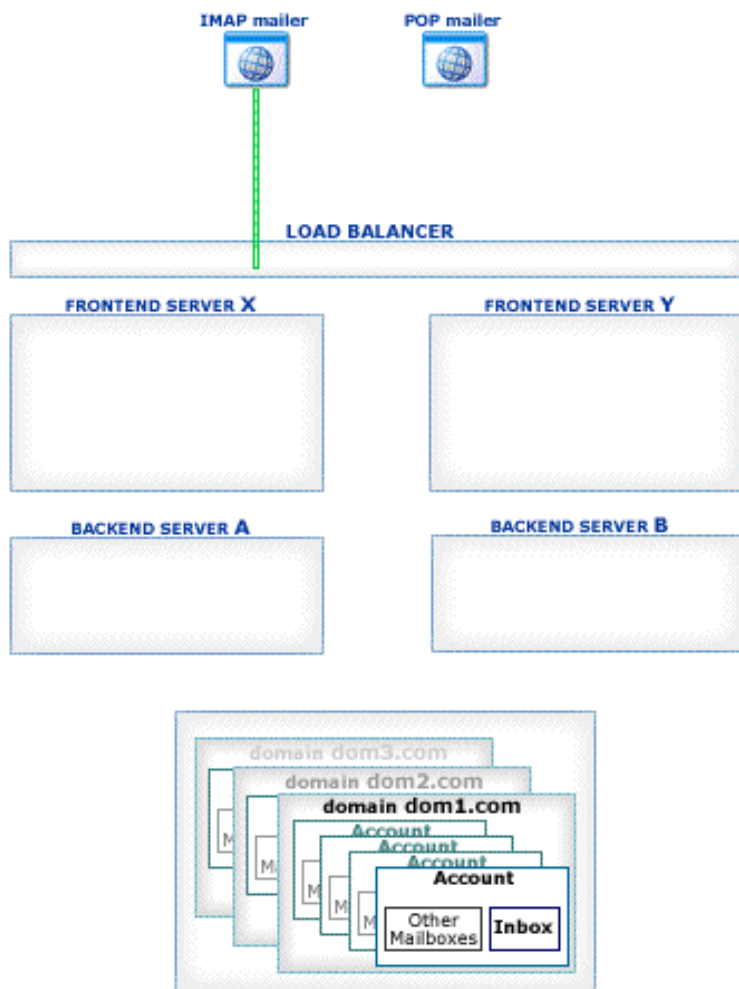
[POP](#), [IMAP](#), [MAPI](#), [ACAP](#) Sessions are created on the Backend server that succeeds to open the target Account. These protocols work over the TCP network protocol. When a TCP connection is established to the Server that cannot open the target Account (to a Frontend Server, or to a "wrong" 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:

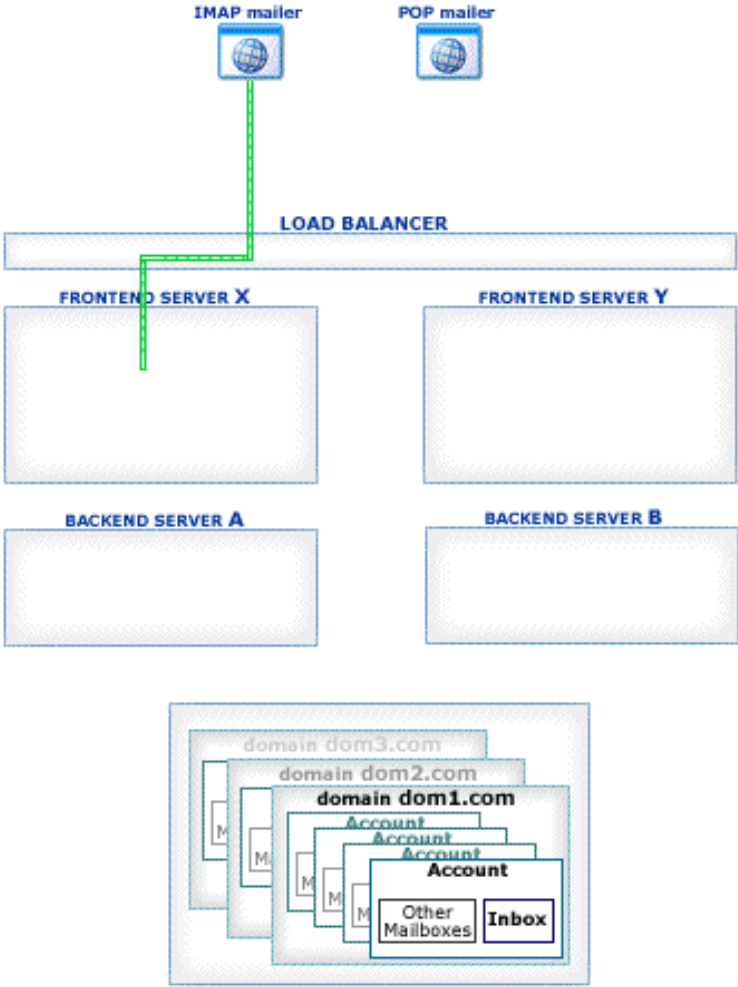
Step 1.



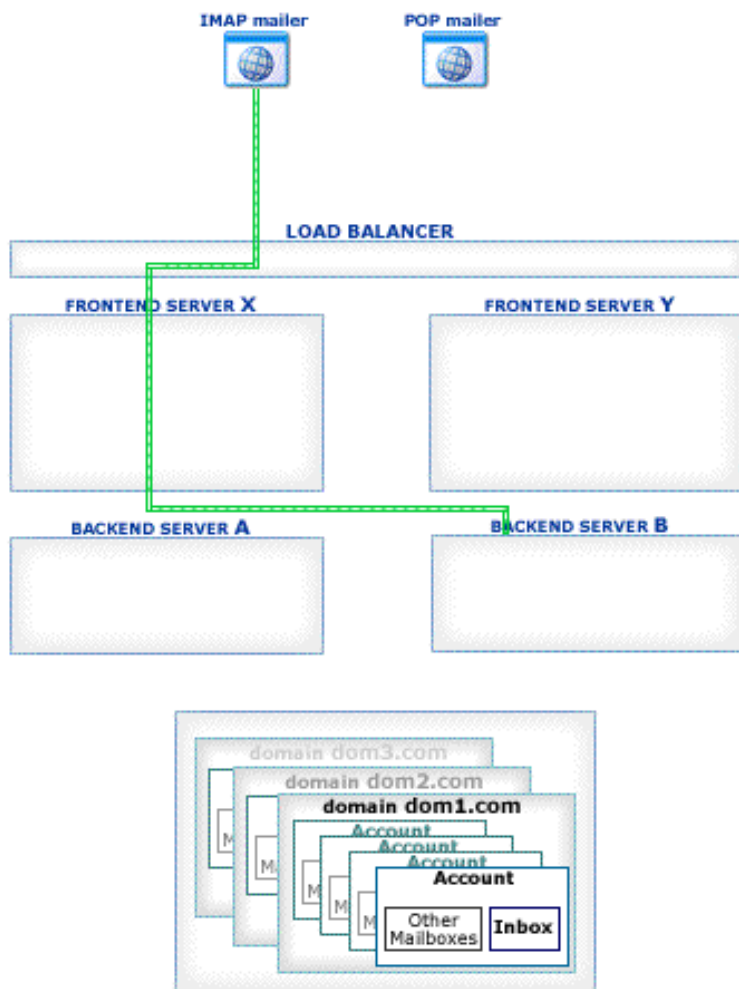
Step 2.



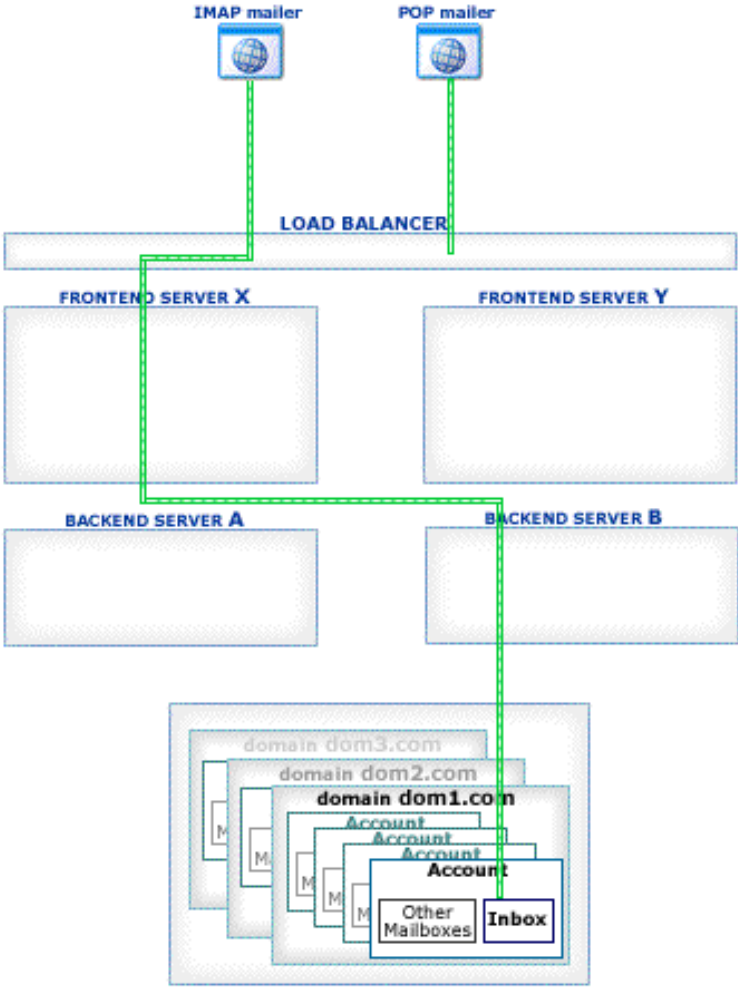
Step 3.



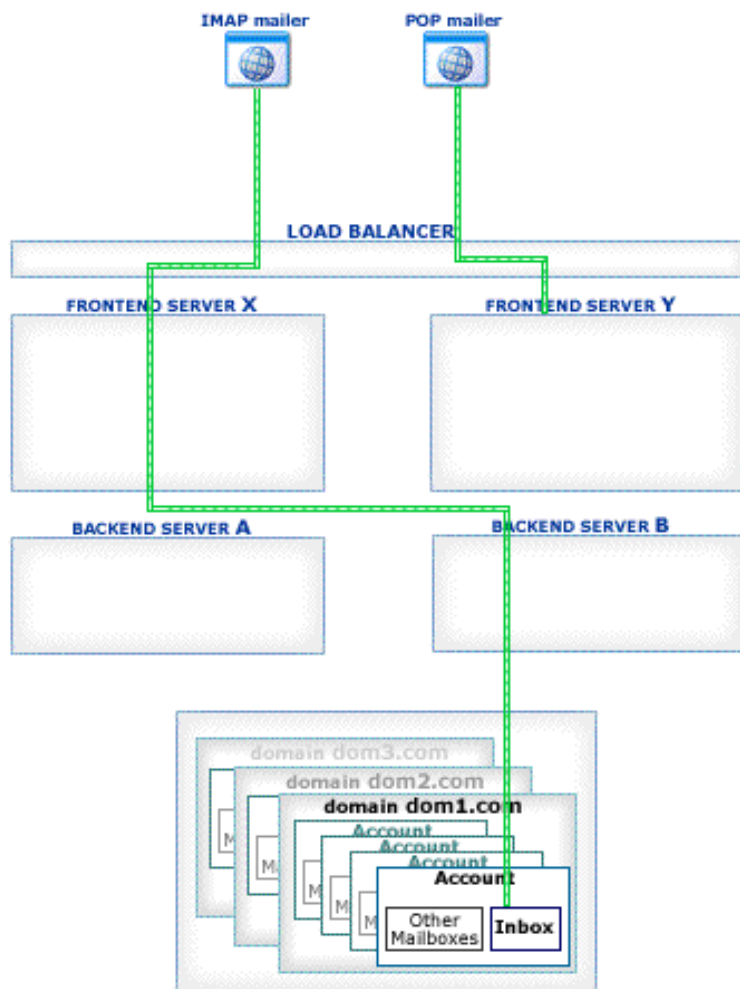
Step 4.



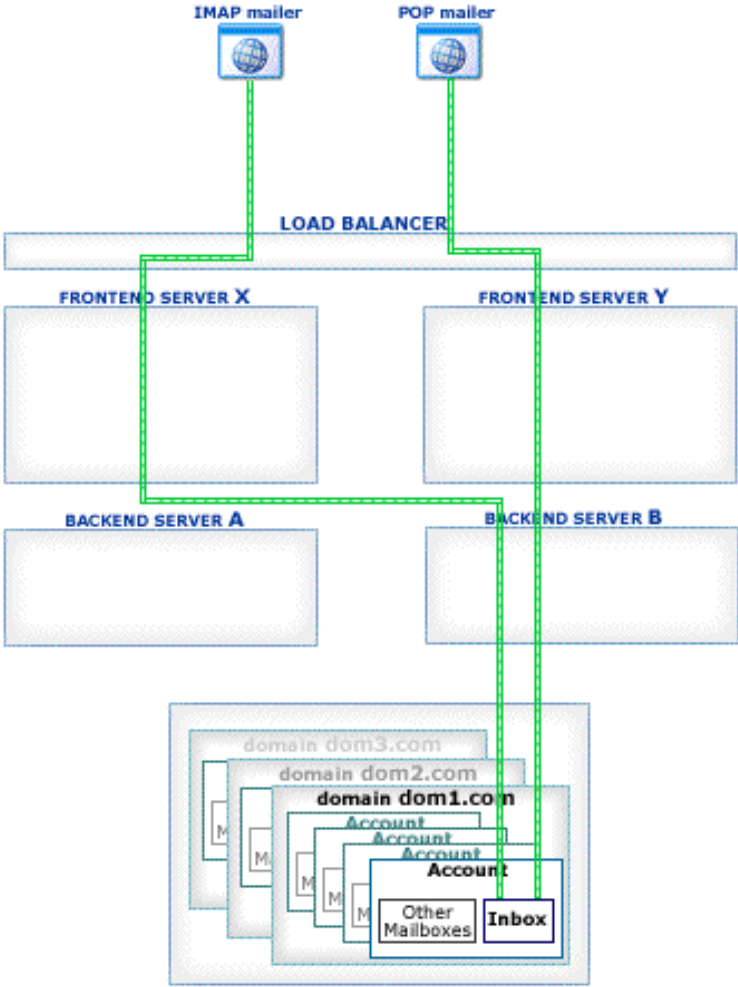
Step 5.



Step 6.



Step 7.



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.

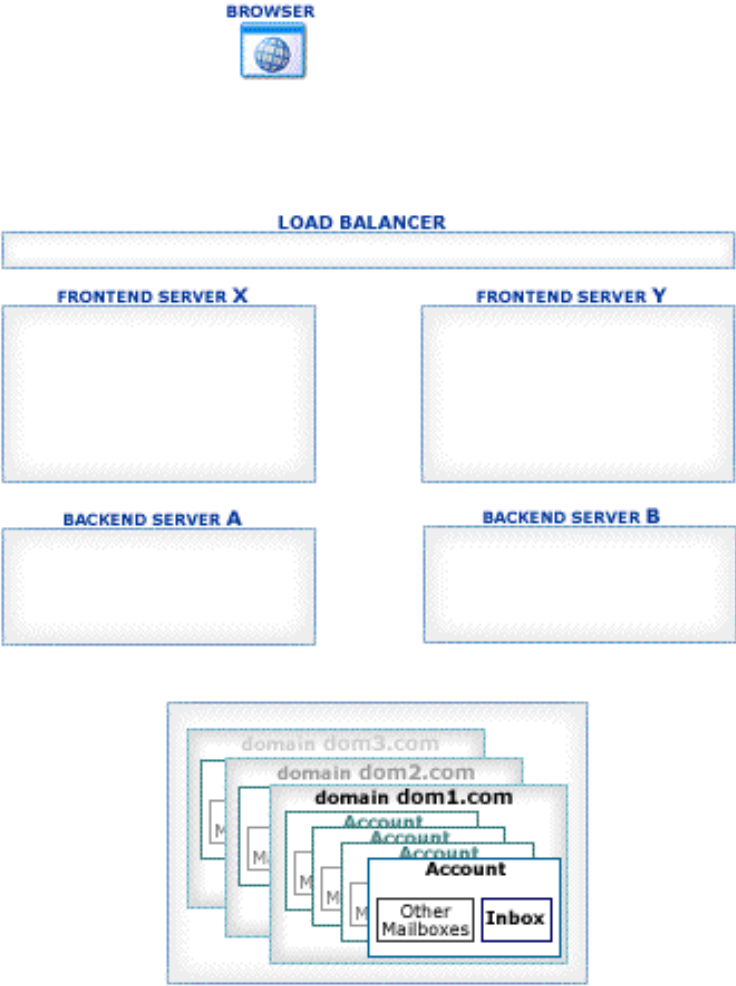
WebUser Interface

[WebUser Interface](#) Sessions are created on the Backend server that succeeds to open the target Account.

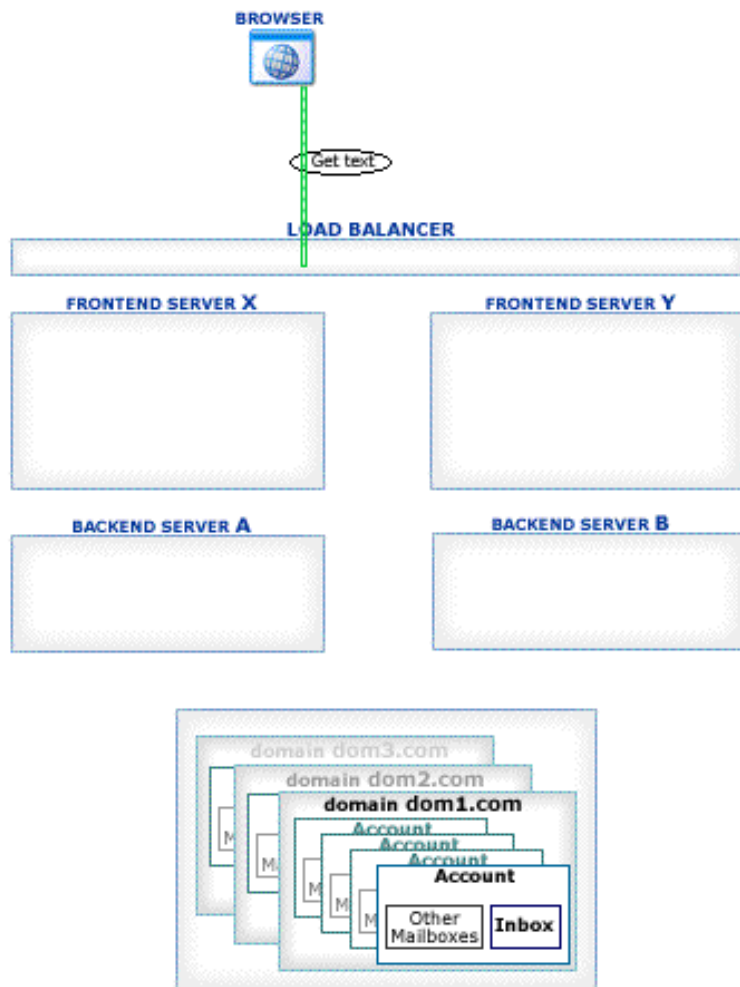
User browsers send HTTP requests to the Frontend Servers via Load Balancer(s). If the session request hits the "wrong" Server (i.e. the server that does not host the target session), the request is proxied to the correct Server.

If the HTTP connection is encrypted (using SSL/TLS), request decryption and response encryption take place on the frontend server:

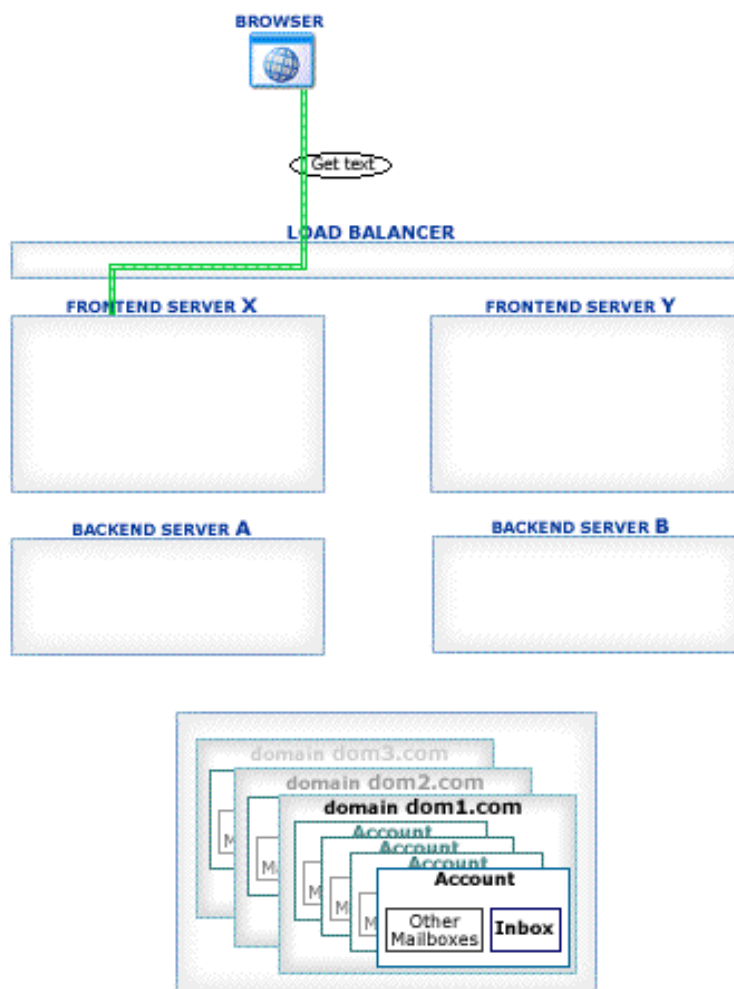
Step 1.



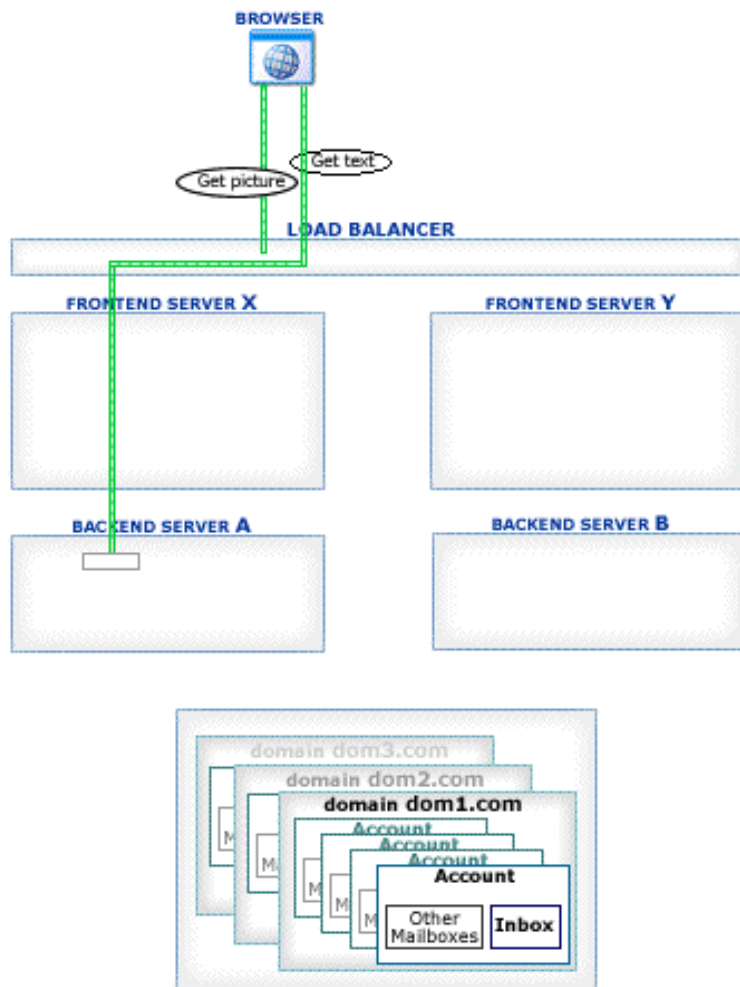
Step 2.



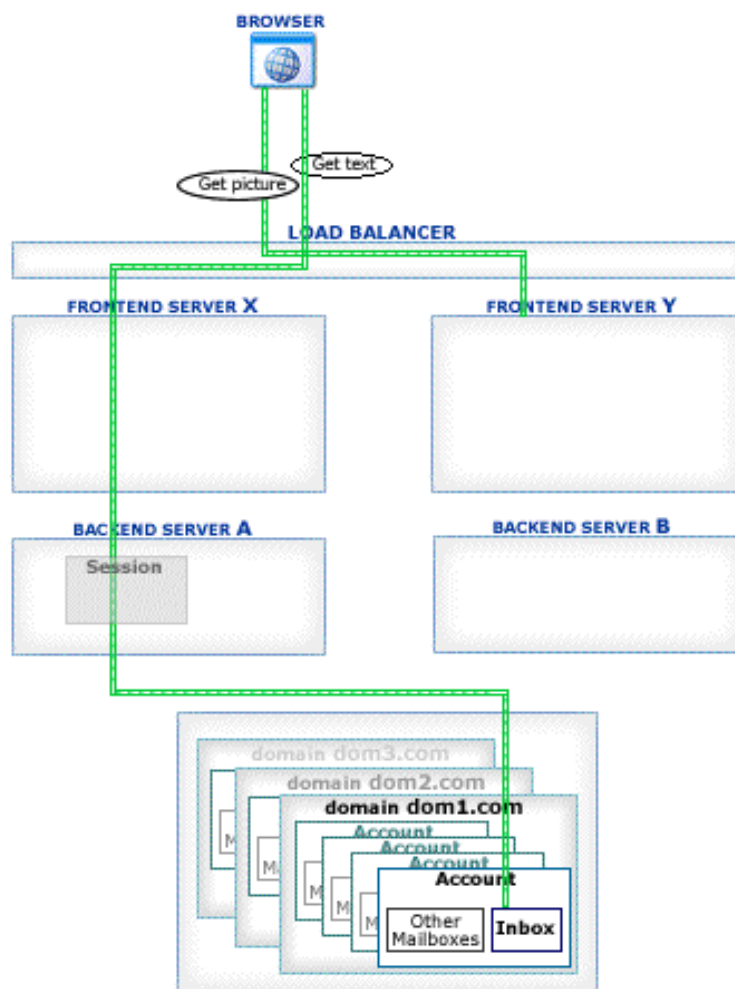
Step 3.



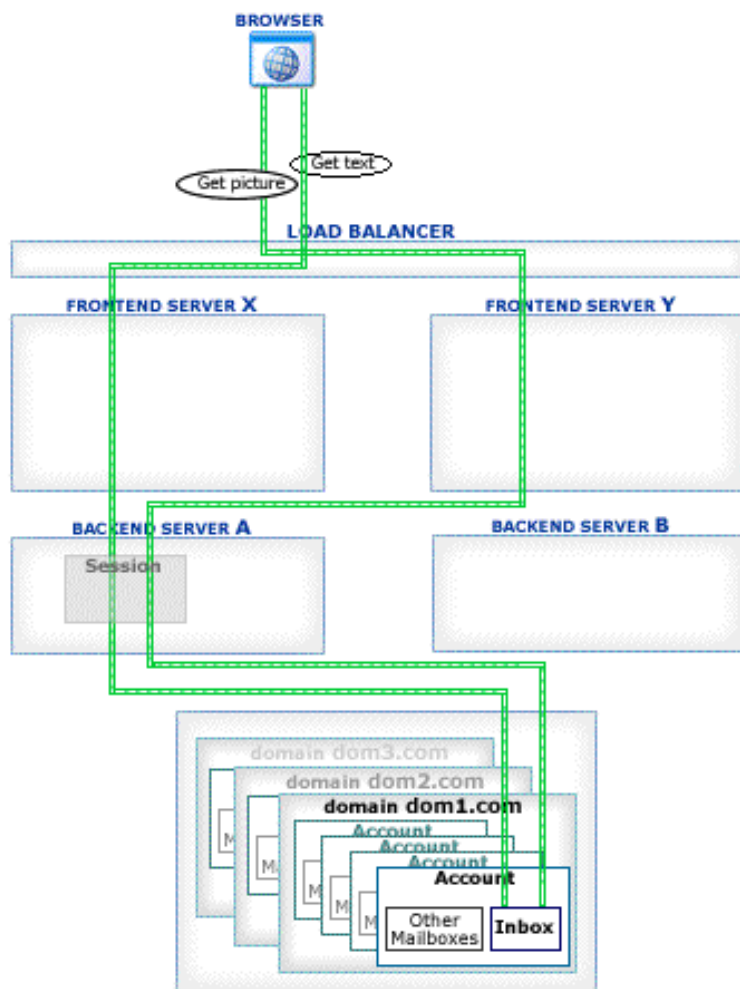
Step 4.



Step 5.



Step 6.





Applications

Many CommuniGate Pro Server 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.



Data Formats

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.

Strings

Strings are basic, unstructured textual data objects.

A textual representation of a string is either an *atom* - a sequence of latin letters and digits, or a *quoted string* - a sequence of any printable symbols except the quotation mark and the backslash symbol enclosed into the quota-

tion 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` and `\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"`

DataBlocks

DataBlocks 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

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 1 or more decimal digits.

Example: #-234657

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: #T22-10-2005_15:24:45

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.

Example: #I10.0.44.55:25

Arrays

An array object is an ordered set of objects (array elements).

An 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:

```
()
```

Dictionaryes

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:

```
{ }
```

Syntax Rules

Below is the formal syntax for textual representations of the fundamental type objects.

```
d-digit      ::= 0 .. 9  
a-symbol    ::= A .. Z | a .. z | d-digit  
l-symbol    ::= a-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  
timestamp   ::= # T day - month - year [ _ hour : minute : second ]  
array       ::= ( [ object 0*( , object ) ] )  
dictionary ::= { 0*( string = object ; ) }
```

```
object ::= string | datablock | number | timestamp |  
array | dictionary
```



Command Line Interface (API)

The CommuniGate Pro Server provides a Command Line Interface (CLI) for server administrating. This interface can be used as an alternative for the standard 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.

The CommuniGate Pro Server provides several methods to access its CLI.

The [CommuniGate Pro Perl Interface](#) document contains the set of the [Perl language](#) utilities that allow a Perl script to access the CommuniGate Pro CLI API. 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. The document also contains links to several sample Java programs.

Administrating the Server via the PWD module

The CommuniGate Pro Server CLI is available as an extension to the [PWD](#) protocol.

As soon as a PWD user is authenticated, the CLI commands are accepted. For each CLI command the server checks the access rights of the authenticated user.

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.

Here is a sample PWD session with CLI commands:

```
C: telnet servername.com 106
S: 200 CommuniGate Pro at mail.stalker.com PWD Server 3.5 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: QUIT
S: 200 CommuniGate Pro PWD connection closed
```

CLI Syntax

The CommuniGate Pro CLI uses the [Dictionary Format](#) 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 Dictionary format is multi-line, all 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.

Account Administration

A user should have the [Account Settings access right](#) or the [Domain Administration access right](#) to use the Account Administration CLI commands.

`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.

`CreateAccount accountName [accountType] [external] [settings]`

Use this command to create new accounts.

accountName : *string*

This parameter specifies the name for the new account.

The name can contain the @sign 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

This optional parameter specifies the type of the account to create. If no account type is specified a MultiMailbox-type account is created.

`external`

This optional flag tells the system to create an account with an external (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 a domain administrator only if the domain administrator has the Can-CreateAccounts access right.

If this command is used by a domain administrator, it will use only those account settings that the domain administrator is allowed to modify.

`RenameAccount oldAccountName into newAccountName`

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).

This command can be used by a domain administrator only if the domain administrator has the `Can-CreateAccounts` 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 a domain administrator only if the domain administrator has the `Can-CreateAccounts` 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 sign (*) instead of an account name. This will indicate the current authenticated account.

Note: All users can send the `GetAccount` 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 that the domain 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 sign (*) instead of an account name. This will indicate the current authenticated account.

Note: All users can send the `GetAccountEffectiveSettings` command for their own accounts.

`SetAccountPassword` *accountName* *To newPassword*

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 is used to specify the new account password. 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.

`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.

`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 a domain administrator only if the domain administrator has the `Can-CreateAliases` access right.

`GetAccountTelnums` *accountName*

Use this command to get the list of phone 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* *newAliases*

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).

newAliases : *array*

This array should contain the telephone number strings. All old numbers assigned to the Account are removed.

This command can be used by a domain administrator only if the domain administrator has the `Can-`

CreateTelnums access right.

GetAccountRules *accountName*

Use this command to get the list of account Rules. The command produces an output - an *array* of the 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).

SetAccountRules *accountName newRules*

Use this command to set the account 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 Rules. All old account Rules are removed.

This command can be used by a domain administrator only if the domain administrator has the Rule-sAllowed access right.

GetAccountRPOP *accountName*

Use this command to get the list of account RPOP records. The command produces an output - an *array* of the 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).

SetAccountRPOP *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 : *array*

This array should contain the account RPOP records. All old account RPOP records are removed.

This command can be used by a domain administrator only if the domain administrator has the Can-ModifyRPOP access right.

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 sign (*) 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 accounts "info" data are stored in .info dictionary files, the "info" elements have dictionary names starting with the hash sign. You should NOT include the hash sign into the key-Name parameter of the GetAccountInfo command.

Sample:

```
GetAccountInfo "user1@domain1.com" (LastLogin LastLoginAd-
dress)
```

Note: the "info" element names are case-sensitive.

The output is a dictionary will all all those "info" elements that exist and are specified in the key-List 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 accounts "info" data are stored in .info dictionary files, the "info" elements have dictionary names starting with the hash sign. You should NOT include the hash sign into the key-Name parameter of the GetAccountInfo command.

Sample:

```
GetAccountInfo "user1@domain1.com" Key LastLogin
```

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.

`GetWebUser` *accountName*

Use this command to get the Account WebUser Settings. The command produces an output - a *dictionary* with the Account WebUser Settings.

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 `GetWebUser` command to retrieve their own WebUser settings.

`SetWebUser` *accountName* *newSettings*

Use this command to set the Account WebUser 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 should contain the new account WebUser Settings. All old Account WebUser Settings are removed.

This command can be used by a domain administrator only if the domain administrator has the `WebUserSettings` access right.

`GetEffectiveWebUser` *accountName*

Use this command to get the effective Account WebUser Settings. The command produces an output - a *dictionary* with Account WebUser Settings. 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 the `GetEffectiveWebUser` command to retrieve their own effective WebUser settings.

`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.

Group Administration

A user should have the [Can Modify All Domains and Account Settings](#) 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 @sign 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 a domain administrator only if the domain administrator has the `Can-CreateGroups` 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 a domain administrator only if the domain administrator has the Can-CreateGroups 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 a domain administrator only if the domain administrator has the Can-CreateGroups 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.

Forwarder Administration

A user should have the [Can Modify All Domains and Account Settings](#) 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 @sign 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 mail to.

This command can be used by a domain administrator only if the domain administrator has the `Can-CreateForwarders` 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 a domain administrator only if the domain administrator has the `Can-CreateForwarders` 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 mail to.

forwarderName : string

This parameter specifies the name of an existing forwarder. The name can include the domain name (see above).

Domain Administration

A user should have the [Can Modify All Domains and Account Settings](#) 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 that the domain administrator is allowed to modify.

`GetWebUserDefaults [domainName]`

Use this command to get the default account WebUser Interface 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.

`SetWebUserDefaults [domainName] newSettings`

Use this command to change the Default WebUser Interface 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 authenticated user Domain.

newSettings : *dictionary*

This dictionary is used to replace the domain Default WebUser Interface settings. All old Default WebUser Interface settings are removed.

This command can be used by a domain administrator only if the domain administrator has 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.

`GetDomainRules` *domainName*

Use this command to get the list of Domain Rules. The command produces an output - an *array* of the Rules specified for the domain.

domainName : string

This parameter specifies the name of an existing domain.

`SetDomainRules` *domainName newRules*

Use this command to set the account Rules.

domainName : string

This parameter specifies the name of an existing domain.

newRules : array

This array should contain the Domain Rules. All old Domain Rules are removed.

This command can be used by a domain administrator only if the domain administrator has the `Rule-sAllowed` 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.

`InsertDirectoryRecords` *domainName*

Use this command to insert records for Domain objects (accounts, groupes, 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 a domain administrator only if the domain administrator has 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 a domain administrator only if the domain administrator has the `CentralDirectory` access right.

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.

`GetAllAccountsDefaults`

Use this command to get the server-wide Default Account settings. The command produces an output - a *dictionary* with the global default account settings.

`UpdateAllAccountsDefaults 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.

`SetAllAccountsDefaults 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]AllAccountsDefaults` commands to work with the cluster-wide Default Account settings.

`GetServerWebUserDefaults`

Use this command to get the server-wide Default WebUser Interface settings. The command produces an output - a *dictionary* with the default settings.

`SetServerWebUserDefaults newSettings`

Use this command to change the server-wide Default WebUser Interface settings.

newSettings : *dictionary*

This dictionary is used to replace the server-wide Default WebUser Interface settings. All old server-wide Default WebUser Interface settings are removed.

`GetClusterWebUserDefaults`

`SetClusterWebUserDefaults newSettings`

These commands are available in the Dynamic Cluster only.

Use these commands instead of the [Get|Set]ServerWebUserDefaults commands to work with the cluster-wide Default WebUser Interface settings.

CreateDomain *domainName* [*settings*]

Use this command to create a new secondary domain.

domainName : *string*

This parameter specifies the domain name to create.

settings : *dictionary*

This optional parameter specifies the domain settings.

RenameDomain *oldDomainName* into *newDomainName*

Use this command to rename a domain.

domainName : *string*

This parameter specifies the name of an existing secondary domain.

newDomainName : *string*

This parameter specifies the new domain name.

DeleteDomain *oldDomainName* [*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 accounts and mailing lists will be removed.

CreateSharedDomain *domainName* [*settings*]

Use this command to create a new shared secondary domain in a Dynamic Cluster.

domainName : *string*

This parameter specifies the domain name to create.

settings : *dictionary*

This optional parameter specifies the domain settings.

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 CommuniGatePro Server.

This operation is allowed only when the Directory-based Domains are enabled.

`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.

`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.

`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.

`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.

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.

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).

Mailbox Administration

A user should be the mailbox owner, or should have the [Can Modify All Domains and Account Settings](#) 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 sign (*) can be used to specify the current authenticated account.

filter : *string*

This optional parameter specifies the filter string to apply to account mailboxes. 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 [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 sign (*) 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 the operation should be executed. If this name is specified, the mailbox is created 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 create the **Archive/March** 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 sign (*) 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 sign (*) 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 created 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 sign (*) 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 sign (*) 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 sign (*) 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 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 sign (*) 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 sign (*) 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.

GETACCOUNTSUBSCRIPTION *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 sign (*) can be used to specify the current authenticated account.

SETACCOUNTSUBSCRIPTION *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 sign (*) 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 sign (*) can be used to specify the current authenticated account.

SETMAILBOXALIASES *accountName newAliases*

Use this command to set the Account "subscribed mailboxes" list.

accountName : string

This parameter specifies the name of an existing account. The asterisk sign (*) can be used to specify the current authenticated account.

newAliases : dictionary

The set of new mailbox aliases.

Alert Administration

A user should have the [Can Modify All Domains and Account Settings](#) 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 sign (*) 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 sign (*) 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 sign (*) 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 sign (*) 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. In order to retrieve files from the `private` directory and its subdirectories, the authenticated user should be the Account owner, or the authenticated user should have the [Can Modify All Domains and Account Settings](#) access right or the WebSite [Domain Administration access right](#).

GETWEBFILE *accountName* FILE *fileName* [OFFSET *position*] [SIZE *sliceSize*]

Use this command to retrieve a file from the account Personal File Site. This command produces an output - a *array of 3 strings*. The first string contains the base64-encoded content of the specified file, the second string contains the file modification date (in the ACAP time format), and the third string contains the current file size.

accountName : *string*

This parameter specifies the name of an existing account. The asterisk sign (*) can be used to specify the current authenticated account.

fileName : *string*

This parameter specifies the name of the Personal File Site file to be retrieved.

position : *number*

If this parameter is specified the File Site 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.

The authenticated user should be the account owner, or should have the [Can Modify All Domains and Account Settings](#) access right or the WebSite [Domain Administration access right](#) to use the Personal File Site Administration CLI commands.

`PUTWEBFILE accountName FILE fileName [OFFSET position] DATA encodedData`

Use this command to store a file in the Account Personal File Site. If a Personal File Site file with the specified name already exists, the old file is removed.

accountName : string

This parameter specifies the name of an existing Account. The asterisk sign (*) can be used to specify the current authenticated Account.

fileName : string

This parameter specifies the name for the Personal File Site file.

position : number

If this parameter is specified and it is not zero, the File Site 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.

encodedData : string

This parameter contains the Base64-encoded content of the Personal File Site file.

`RENAMEWEBFILE accountName FILE oldFileName INTO newFileName`

Use this command to rename a file in the Account Personal File Site.

accountName : string

This parameter specifies the name of an existing Account. The asterisk sign (*) can be used to specify the current authenticated Account.

oldFileName : string

This parameter specifies the name of an existing Personal File Site file.

newFileName : string

This parameter specifies the new name for the Personal File Site file.

`DELETEWEBFILE accountName FILE fileName`

Use this command to remove a file from the account Personal File Site.

accountName : string

This parameter specifies the name of an existing Account. The asterisk sign (*) can be used to specify the current authenticated Account.

oldFileName : string

This parameter specifies the name of an existing Personal File Site file.

`LISTWEBFILES accountName [PATH filePath]`

Use this command to list all files in the Personal File Site top directory or in one of its subdirectories.

This command produces an output - a *dictionary*, where each key is a name of the File Site file, and the key value is a *dictionary* for regular file and empty *array* for subdirectories.

accountName : string

This parameter specifies the name of an existing Account. The asterisk sign (*) can be used to specify the current authenticated Account.

filePath : string

This optional parameter specifies the name of the Personal File Site subdirectory. You can omit this parameter along with the PATH keyword, in this case the command returns the list of files in the top Personal File Site directory.

`GETWEBFILESINFO accountName`

Use this command to get the statistical information about all files in the Personal File Site area. This command produces an output - an *array* with 2 *string* elements. The first element contains the total size of all File Site files, the second element contains the number of files in the File Site area.

accountName : string

This parameter specifies the name of an existing Account. The asterisk sign (*) can be used to specify the current authenticated Account.

Mailing Lists Administration

A user should have the [Accounts And Domains Server access right](#) or the [Domain Administration 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 - an *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) 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).

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 an existing mailing list. 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. 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 Accounts and Domains access right.

`GetList listName`

Use this command to retrieve list settings. The command produces an output - a *dictionary* with the list-Name 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* 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 for 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 [Account Settings 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.

RenameDomainSkin [*domainName* SKIN] *skinName* INTO *newSkinName*

Use this command to rename 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 an existing 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.

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 string with the BASE64-encoded Skin file content, the second array element is a string with the file modification date in the ACAP date format.

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 : *string*

This string contains the BASE64-encoded 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 Domain 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 string with the BASE64-encoded Skin file content, the second array element is a string with the file modification date in the ACAP date format.

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 : string

This string contains the BASE64-encoded 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.

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 [ 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/rrrrrrrrrrrr/Mailboxes.wssp` where *rrrrrrrrrrrr* is the Session ID string returned.

account : *string*

This parameter specifies the Account name.

ip-address : *string*

This parameter specifies the IP address of the client browser. If the Account has the "Fixed IP" WebUser Preference setting enabled, connections to the session will be allowed from that IP address only.

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 [Can Modify All Domains and Account Settings](#) access right or the CanCreateWebUserSessions [Domain Administration access right](#) to create WebUser Sessions.

```
FindWebUserSession accountName [ ADDRESS ip-address ]
```

Use this command to find an existing WebUser session for the specified account. The command produces an output - a *string* that contains the WebUser Session ID.

account : *string*

This parameter specifies the Account name.

ip-address : *string*

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" WebUser Preference setting disabled or have the same login IP address as the specified one.

The authenticated user should have the [Can Modify All Domains and Account Settings](#) access right or

the CanCreateWebUserSessions [Domain Administration access right](#) to use this command.

```
GetWebUserSession sessionID [ DOMAIN domainName ]
```

Use this command to retrieve the WebUser 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 WebUser Session ID.

domainName : *string*

This optional parameter specifies the name of Domain the session Account belongs to.

The authenticated user should have the [Can Modify All Domains and Account Settings](#) access right to retrieve WebUser 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 WebUser session inactivity timer.

```
KillWebUserSession sessionID [ DOMAIN domainName ]
```

Use this command to terminate a WebUser Session.

sessionID : *string*

This parameter specifies the WebUser Session ID.

domainName : *string*

This optional parameter specifies the name of Domain the session Account belongs to.

The authenticated user should have the [Can Modify All Domains and Account Settings](#) access right to terminate a WebUser 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.

Real-Time Application Administration

The following commands can be used to manage CommuniGate Pro [Real Time](#) Application Environments.

A user should have the [Account Settings 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 optional parameter specifies the domain name. If the domain name is not specified, the command applies to the administrator domain.

language : *language*

This optional 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 : *language*

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*

StoreServerSkinFile *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.

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 : *language*

This optional 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 ]  
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.

Server Settings

A user should have the [Server Settings access right](#) to use the Server Settings CLI commands.

```
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.

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.

GetBlacklistedIPs

Use this command to retrieve the set of Blacklisted IP Addresses. The command produces an output - a (multi-line) *string* with Blacklisted IP addresses and address ranges.

GetClientIPs

Use this command to retrieve the set of Client IP Addresses. The command produces an output - a (multi-line) *string* with Client IP addresses and address ranges.

GetWhiteHoleIPs

Use this command to retrieve the set of WhiteHole IP Addresses. The command produces an output - a (multi-line) *string* with White Hole IP addresses and address ranges.

GetProtection

Use this command to retrieve the Protection settings. The command produces an output - a *dictionary* with the server Protection 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.

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.

SetBlacklistedIPs *newAddresses*

Use this command to update the set of Blacklisted IP Addresses.

newAddresses : string

This (multi-line) string parameter contains the set of addresses and address ranges forming the new set of Blacklisted IP Addresses.

SetClientIPs *newAddresses*

Use this command to update the set of Client IP Addresses.

newAddresses : string

This (multi-line) string parameter contains the set of addresses and address ranges forming the new set of Client IP Addresses.

SetWhiteHoleIPs *newAddresses*

Use this command to update the set of WhiteHole IP Addresses.

newAddresses : string

This (multi-line) string parameter contains the set of addresses and address ranges forming the new set of WhiteHole Addresses.

SetProtection newSettings

Use this command to set the server Protection Settings.

newSettings : dictionary

New server Protection settings.

SetBanned newSettings

Use this command to set the server Banned Message Line Settings.

newSettings : dictionary

New server Banned settings.

GetClusterLANIPs

GetClusterBlacklistedIPs

GetClusterClientIPs

GetClusterWhiteHoleIPs

GetClusterProtection

GetClusterBanned

SetClusterLANIPs newAddresses

SetClusterBlacklistedIPs newAddresses

SetClusterClientIPs newAddresses

SetClusterWhiteHoleIPs newAddresses

SetClusterProtection newSettings

Use these commands to retrieve and update the Cluster-wide IP Address lists and Protection settings.

GetServerRules

Use this command to read the Server-Wide Automated Mail Processing Rules. The command produces an output - an *array* of the Server Rules.

SetServerRules newRules

Use this command to set the Server-Wide Automated Mail Processing Rules.

newRules : array

An array of new Server Rules.

GetClusterRules

Use this command to read the Cluster-Wide Automated Mail Processing Rules. The command produces an output - an *array* of the Cluster Rules.

`SetClusterRules newRules`

Use this command to set the Cluster-Wide Automated Mail Processing Rules.

newRules : *array*

An array of new Cluster 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`

These commands are the same as the `GetRouterTable` and `SetRouterTable` commands, but they deal with the Cluster-Wide Router Table.

`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.

RefreshOSData

Use this command to set 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.

A user should have the [Server Settings access right](#) or the [Account Settings access right](#) 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.)

Monitoring

A user should have the Monitoring [access right](#) to use the Server Monitoring CLI commands.

`GetSNMPElement ObjectID`

Use this command to retrieve the current value of a server state (SNMP) element.

ObjectID : string

The object ID of the server state element (see the [SNMP](#) section for more details).

This command produces an output - a *string* with the server state element value.

Shutdown

Use this command to stop the CommuniGatePro Server.

Access Rights Administration

A user should have the [unlimited 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 Account and Domains](#) administration access right.

Statistics

The Account-Level Statistics data is collected if the Account Statistics option is enabled on the Obscure page in the Settings realm of the CommuniGate Pro WebAdmin Interface.

`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 sign (*) 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 *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.

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 sign (*) 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

```
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 sign (*) 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 sign (*) 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
BytesReceived	The total size of all messages delivered to the Domain

Miscellaneous Commands

```
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 [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.

To use this command, the user should have the "Can Reject Queues" Server Administration right.

`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.

`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.

`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.

To use this command, the user should have the "Can Monitor" Server Administration right.

`SetTempBlacklistedIPs addresses`

addresses : number

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.

`RemoveAccountSubset accountName SUBSET subsetName`

Use this command to remove an Account "dataset" (such as RepliedAddresses dataset).

accountName : string

This parameter specifies the name of an existing account. The asterisk sign (*) 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.

To use this command on other user subsets, the user should have the Domain Administration right.



CommuniGate Programming Language (CG/PL)

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 Signal](#) control.
- [Real-Time Applications](#) (such as "IP-PBX").

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.

Language Elements

The following is a simple CG/PL program:

```
//  
// A simple CG/PL application  
//  
entry Main is  
  myName = "Jim" + " " + "Smith";  
  if length(myName) > 10 then  
    myName = Substring(myName,0,8) + "..";  
  end if;  
end;
```

Data Model

Information is presented as *objects*. An object can be a [string](#), a [number](#), a [datablock](#), a [timestamp](#), an [ip-address](#), an [array](#), or a [dictionary](#). 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 a *null-value* (null value or "no object" value).

When processing logical (*boolean*) values, a *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**, **elif**, **else**, **entry**, **exitif**, **external**, **false**, **forward**, **function**, **if**, **is**, **not**, **loop**, **null**, **or**, **procedure**, **return**, **spawn**, **stop**, **then**, **true**, **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.

- *numbers* - sequences of digits.
- *strings* - quoted strings as specified in the [Data](#) section.
- *signs* - +, -, *, //, /, %, =, ==, !=, <, <=, >, >=, !, &, &&, |, ||, ?, :, ;, ,, :, (,), [,], {, }.

A comment is a double-slash (//) sign and all following symbols up to and including the next EOL (end-of-line) symbol.

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.

Variables

Any name not declared as a [procedure or function](#) name, and not matching a [built-in](#) procedure or function name is a name of a variable.

Each variable has a value assigned to it. Initially, this value is a null-value. Any object can be assigned to any variable to become that variable value.

```
myCount = 3;           // the myCount variable is created and
                        // a 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
```

Variables are local to the function/procedure instance (invocation) where they were created. All their values are destroyed when the function/procedure exits (returns).

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 **!** sign 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 calculated 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 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.

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 numeric operands, to produce a true-value when the arithmetic comparison condition is met.

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 block position.
- Timestamps are equal if they specify the same second.
- 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, every 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 **&** sign 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 **|** sign 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 **^** sign 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 **&&** sign 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 **||** sign 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: $X \text{ op } Y \text{ op } Z$ is the same as $(X \text{ op } Y) \text{ op } Z$.

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 its 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"
4 > 5 ? 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 or a dictionary value (otherwise the operation results in a program exception), and

the *index* expression should have a numeric value smaller than the number of *object* elements.

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]
```

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*).

The name used 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(1, "zzz") [1]
```

The keywords **null** and **false** can be used in expressions to specify a null-value, while the keyword **true** can be used to specify a true-value.

Operators

An operator sequence contains zero or more operators, followed by the semicolon (';') sign.

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 = sign and an expression that will be computed and assigned to the specified data container.

```
myVar = 123 + 111;  
myVar = "string";
```

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(1, "zzz");
```

A stop operator terminates the Task execution. It consists of the **stop** keyword:

```
stop;
```

If there is an active connection, that connection is closed.

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 the **loop** keyword, optionally prefixed with the **while** keyword and an expression (*while-expression*), an operator sequence (*initial sequence*), and the **end** keyword optionally followed by the **loop** keyword.

If a while-expression is specified, it is computed, and if its value is null the loop operator execution ends. Otherwise the operator sequence is executed, and the loop operator execution repeats.

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 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 (`;`) sign, 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 = myWord + "aaa";
    exitif length(myWord) >= 20;
    myWord = 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 sign.

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 signs.

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 loop operator consists of the **while** keyword, an expression (*while-expression*), a left brace ('{ ') sign, an operator sequence (*initial sequence*), zero or more exitif-portions and a right brace (' } ') sign.

Each optional exitif-portion consists of the **exitif** keyword, followed by an expression (exitif-expression), the semicolon (' ; ') sign, 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.

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 *reenterabile*: 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 should contain exactly one external-declaration or exactly one definition (but not both) of each code section used.

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.

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](#) environments. See the environment description for more details.

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.

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 (;) sign

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 (;) sign.

for an alternative entry definition:

the left brace ({) sign followed by an operator sequence, followed by the right brace (}) sign.

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 (;) sign

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 (;) sign.

for an alternative entry definition:

the left brace ({) sign followed by an operator sequence, followed by the right brace (}) sign.

for an external-declaration:

the **external** keyword followed by the semicolon (;) sign

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,units) 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 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.

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")` value 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.

`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.

`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.

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 num-

ber.

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 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.

`FindSubstring(str, substr)`

If the *str* value is a string, and the *substr* value is a string, and the latter is a substring of the *str* value, this function returns the position of the substring *substr* in the *str* string (positions start with 0).

For example, the value of `FindSubstring("forest", "for")` is 0.

In all other cases this function returns the number -1.

`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 the *from* is non-negative, 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, *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.

`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.

`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`.

TimeStamps

`IsDate (arg)`

This function returns a true-value if the *arg* value is a timestamp, otherwise the 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.

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](#)).

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 (the number 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 mid-night of the specified date.

The following expression timestamp value is midnight, 05-Nov-2006:

`DateByMonthDay (2006, 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.

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 not an array, a negative numeric value is returned.

`RemoveElement(target, index)`

This procedure removes an element from 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 the array element to remove. In other cases the first (the zero index) element is removed.

If the *myArray* value is `(1, 4, 9, 16, 25)`, the `RemoveElement(myArray, 2)` changes the *myArray* value to `(1, 4, 16, 25)`.

`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")` changes the *myArray* value to `(1, 4, "Jack", 9, 16, 25)`.

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.

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 a null-value.

`TextToObject(arg)`

The *arg* value should be a string or a datablock, otherwise this function call results in a program exception.

This function returns an object textually represented by the *arg* value. If conversion fails, the function

returns a null-value.

Environment

`Vars()`

This function returns a dictionary unique for this *Task* (a program invocation). This dictionary can be used to store variables visible in all procedures and functions.

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 Email 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.

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* is not a non-empty string, the *keyValue* value must be a dictionary. It is used to update the Account Preference Data.

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.

This function returns a null-value if Preference data is successfully updated, otherwise it returns a string with an error code.

`ReadSiteFile (fileName)`

This function reads a file from the current Account [File Storage](#).

The *fileName* value should be a string. It specifies the name of the file to read.

This function returns a datablock value with the file content or a null-value if the specified file is not found.

`WriteSiteFile (fileName, data)`

This function stores the *data* datablock or string into the *fileName* file in the current Account [File Storage](#).

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 was successfully written, otherwise it returns a string with an error code.

`AppendSiteFile (fileName, data)`

This function appends the *data* datablock or string to the end of the *fileName* file in the current Account [File Storage](#).

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.

`DeleteSiteFile (fileName)`

This function deletes the *fileName* file from the current Account [File Storage](#).

This function returns a null-value if the file was successfully deleted, otherwise it returns a string with an error code.

`RenameSiteFile (oldFileName, newFileName)`

This function renames the *oldFileName* file in the current Account [File Storage](#) into *newFileName*.

Both parameters must have string values.

This function returns a null-value if the file was successfully renamed, otherwise it returns a string with an error code.

`ListSiteFiles (folderName)`

This function returns information about all files in the specified subdirectory of the current Account [File Storage](#).

If *folderName* is not a string, information about the top-level File Site directory 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 subfolder name. For folders, the dictionary value is an empty array. For

files, the dictionary value is a dictionary with the following elements:

STCreated

a timestamp value with the file creation date.

STModified

a timestamp value with the file modification date.

STFileSize

a numeric value with the file size in bytes.

Note: To access other Accounts File Storage, specify the file or folder name as

~account[@domainName]/fileName

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](#) from the current Domain.

The *groupName* value should be a string. It specifies the name of the Group to read.

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.

Mailbox Handles

Mailbox handles are internal objects representing a [Mailbox](#).

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 was opened successfully, otherwise it returns a null-value.

`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");
```

`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.

`MailboxSetFlagsByUID (boxRef, uid, flags)`

This function modifies mailbox message flags.

The *boxRef* value should be a mailbox handle, the *uid* value should be a number - the message UID, the *flags* value should be a comma-separated list of message flag [Names](#) and/or [Negative Names](#).

The function sets the flags specified by 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, the *uid* value should be a number - the message UID,

`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.

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

`SysLog(arg)`

This procedure places the textual representation of the *arg* value into the Server Log.

`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.

Communications

`HTTPCall(URL, parameters)`

This function performs an HTTP transaction.

The *URL* value should be a string. It specifies the request URL. The URL schema should be `http` or `https`.

The *parameters* value should be a dictionary. It specifies the request parameters and, optionally, the request body. The following dictionary elements are processed (all of them are optional):

body

a datablock with the request body.

Content-Type

a string with the request body content type. This element is used only when a *body* is specified.

Content-Subtype

a string with the request body content subtype. This element is used only when a *body* and **Content-Type** elements are specified.

method

a string with the request method. If this element is absent, the **GET** method is used if no *body* is specified, otherwise the **POST** method is used.

Cookie

a string with the Cookie field data.

authName, authPassword

when present, these strings are used to authenticate the request.

The function returns an error code string if HTTP transaction failed. Otherwise the function returns a dictionary with the following elements:

`responseCode`

a number with the HTTP response code.

`body`

a datablock with the response body (may be absent).

`Content-Type`

a string with the response body content type.

`Content-Subtype`

a string with the response body content subtype.

`charset`

a string with the response body charset.

`Date, Last-Modified, Expires`

timestamp elements with the response header field values.

`Server, Location, Set-Cookie`

string elements with the response header field values.

`SendEmail(fromAddress, subject, to, headers, content)`

This function composes and sends an E-mail message.

The *fromAddress* value should be a string. It specifies the message `From:` address.

The *subject* value should be a string. It specifies the message `Subject:` field.

The *to* value should be an E-mail string or an array of E-mail strings. It specifies the message `To:` address(es).

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):

`Cc`

the element value should be an E-mail string or an array of E-mail strings. It specifies the message `Cc:` address(es).

`Bcc`

the element value should be an E-mail string or an array of E-mail strings. It specifies the message `Bcc:` address(es).

`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.).

`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.

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 Email 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.

```
result = SendEmail("from@sender.dom", "Test Message", "To@recipient.dom", null, "Test Message\\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 = ReadSiteFile(dataPart.fileName);
content[1] = dataPart;

headers = NewDictionary();
headers("Content-Class") = "message";

result = SendEmail("from@sender.dom", "Test Attachment", "To@recipient.dom", content, headers);
```

```
ent.dom", headers, content);
```

```
SendInstantMessage (fromAddress, toAddress, content)
```

This function composes and sends an Instant Message.

The *fromAddress* value should be a string. It specifies the message From: address.

The *toAddress* value should be a string. It specifies the message To: (recipient) address.

The *content* value should be a string. It specifies the message content.

The function only initiates an Instant Message signalling operation, it does not wait for this 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.

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 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 = ReadSiteFile("file" + String(nameIndex));
```

```
    if fileData != null then
        resultCode = WriteSiteFile("backup" + String(nameIndex));
        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.

Events

Tasks can exchange data by sending Events:

`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, and an *eventParam* value should be null or any other "unmodifiable" object - i.e. the *eventParam* value cannot be an array or a dictionary.

If you need to send an array or a dictionary parameter with an Event, send its [textual representation](#) instead.

The function only sends an Event to the specified (target) task. It does not wait till the 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:

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.

what

the string value of the *eventName* parameter used in the SendEvent operation in the sender Task.

parameter

the value of the *eventParam* parameter used in the SendEvent operation in the sender Task.

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 a 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.

`CreateMeeting(setName, key, param)`

This function creates a Meeting object in some Meeting Set within the current Account.

The *setName* parameter specifies the Meeting set name. If the parameter 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.

The *parameter* parameter 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)`

This function adds the current Task task handle 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*.

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)`

This function removes the current Task task handle from a Meeting in the current Account.

The *setName* and *key* parameter values specify an already created Meeting, and that Meeting should contain the current Task task handle.

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 of an *Active Task*, if any.

In a [Cluster](#) environment, the Active Task and the current Task may be running on different Cluster member computers.

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.

`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 *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 *pty* parameter value should be a string containing a decimal number with one digit, the dot (.) sign 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 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 function removes the current Task from the 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. The `position` element will not exist in the returned dictionary.

`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 of 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*

`basicData` ::= *variable* | *funcCall*

`dataRef` ::= *basicData* | *dataRef* [*expr*] | *dataRef* . *name* | *dataRef* . (*expr*)

`spawnExpr` ::= **spawn** *name*

`basicExpr` ::= *string* | *number* | *dataRef* | **null** | **false** | **true** | *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     ::= name ( [argList] )
procCall     ::= name ( [argList] )

leftSide     ::= variable | dataRef [ expr ] | dataRef . name | dataRef . (
expr )
letOper      ::= letOper = 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 } ]

loopOper     ::= [ while expr ] loop opSequence 0*( exitif expr ; opSequence )
end [ loop ]
altLoopOper  ::= while expr { opSequence 0*( exitif expr ; opSequence ) }

oper         ::= nullOper | procCall | letOper | returnOper | stopOper |
ifOper | loopOper |
```

```
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 name ( [ paramlist ] ) procBody
function     ::= function name ( [ paramlist ] ) funcBody
program      ::= 1*(entry | procedure | function)
```



Automated Processing (Rules)

The CommuniGate Pro Server can automatically process [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 messages and to all signals sent to a particular Server 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, open the Queue pages in the Settings section of the WebAdmin Interface and click the Rules link.

To specify Server-Wide Signal Rules, open the RealTime pages in the Settings section of the WebAdmin Inter-

face 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.

Creating, Renaming and Removing Rules

When the list of Rules appears in a browser window, the Rule names and priorities can be modified:

Add Rule			
Priority	Name	Edit	Delete
7	Sports List	Edit	<input type="checkbox"/>
2	From Ali	Edit	<input type="checkbox"/>
disabled	Info Reply	Edit	<input type="checkbox"/>

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 `disabled` priority are not applied to the messages, but they are not deleted from the Account Rules 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 a message 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 message and signal processing Rules:

Time Of Day `[is | is not | less than | greater than] 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:

Time Of Day ▼	is greater ▼	8:15am
Time Of Day ▼	is less ▼	5:15pm

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 only before or after the specified date and time.

A date string should be specified in one of the following formats:

- DD MM YYYY
- DD MM YYYY hh:mm
- DD MM YYYY hh:mm:ss
- DD MM YYYY hh:mm:ss +ZZZZ
- DD MM YYYY hh:mm:ss -ZZZZ

where:

DD is the day of month

MM is month specified as 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 Server time zone is used.

Sample:

Current Date	▼	is greater	▼	20 Jun 2003
Current Date	▼	is less	▼	27 Jun 2003 15:20:00

Current Day [is | is not | in | not in] *day string*

This condition checks the current day of week (using the Server local time zone). This condition allows you to compose rules that are applied to messages only on certain days of week.

Days should be specified either as numbers (0 for Sunday, 6 for Saturday), or as RFC822 abbreviations (Sun, Mon, Tue, Wed, Thu, Fri, Sat).

Sample:

Current Day	▼	in	▼	Sat,Sun
-------------	---	----	---	---------

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	▼	SPAM
------------------	---	----	---	------

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 (#) sign.

For example, the Condition operation

```
Sender      in      #BlockedSenders
```

checks if the message sender's address is included into the String List called `BlockedSenders`.

String List subsets can be used as WebUser Interface [Address Books](#).

Rule Actions

Each Rule can have zero, one, or several actions. If a message meets all the Rule conditions, the Rule actions are performed.

Rule Action parameters can contain Macro Symbols ($\wedge X$, where X is a letter). Different sets of Macro Symbols are processed in [Signal](#) and [Email](#) Rules.

The following Rule actions can be used in message and signal processing 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.

`Reject [error message text]`

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 is rejected with an error code.

If a Rule is an Email 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 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

SendURL [URL]

A 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.

SendIM [message text]

An Instant message with the specified message text is sent.

If the message text starts with the

To: *user@domain*

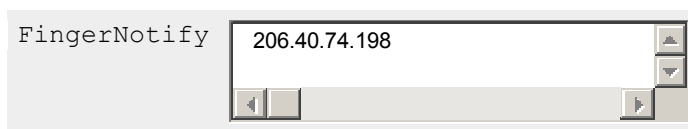
line, then the Instant Message is sent to the specified address. Otherwise, the message is sent to the current Account (for Account-level Rules), or it is not sent at all (for Server-wide/Cluster Rules).

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 should be used with the [NotifyMail®](#) utility installed on client computers.

Sample:



Users are allowed to specify this action only if they are allowed to specify *execute*-type actions.

You can configure the Notifier settings using the Obscure page in the Settings WebAdmin realm.

Write To Log *string*

A Major-Level (Level 2) record with the message ID and the specified string is placed into the [System Log](#).

Only the Server Administrator is allowed to specify this action.

Remember 'From' in *string*

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 (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.



Helper Applications

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 Protocol

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 program-

ming 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 CommuniGatePro *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 7.

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 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 of the addresses 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 Authenticator [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
```

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 3*.

- 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 OK

- If message processing should proceed, but the external filter wishes to add a header field to the message, the response line should have the following format:

seqNum ADDHEADER *header-field-text*

where *header-field-text* is a text string with one or several RFC822 header fields. This (optionally multi-line) text should be placed into the response line in the CommuniGate Pro [String format](#)

- 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

where *seqNum* is the request identifier.

If the program has sent the FAILURE response or any unlisted response, the Server places a record into the 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 1.

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 all 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 dictio-

naries:

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 not passed to the Helper and are ignored in Helper responses:

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 User-Name string attribute is passed to the Helper, but it is ignored in Helper responses.

Accounting requests also have a numeric attribute 0 - the RADIUS protocol request ID. It can be used to detect retransmitted packets (duplicate requests).

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 {1="User1";4=10.0.0.1;32="NAS
1";31=4153837164;} {RealName="User"; NATIP="192.168.1.3";}
O: 00002 ACCEPT {8=192.168.1.3; 9=255.255.255.0; 13=(0,3);}
I: 00002 LOGIN user1@domain1.com {1="uSEr1";32="NAS
2";31=415.5512.12; 8=192.168.1.3;} {NATIP="10.0.1.114";}
O: 00003 REJECT
```



```
I: 00004 ACCNT started user1@domain1.com {0=120;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](#) module when a call is attempted, established, or tiered down.

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 a [text format](#)

when the record is processed, the program should return a positive response:

```
nnnnnn OK
```




Real-Time Application Module

The CommuniGate Pro Real-Time Application module provides an infrastructure to design and deploy applications processing various [Signals](#).

Real-Time Applications to process voice calls include voice mail, auto-attendant, conferencing, and other applications. 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.
- [VoiceXML](#) files with Application code.

- 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 Domains realm of the WebAdmin Interface, and click the PBX link.

To manage the Domain Environment, open that Domain page in the Domains 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:

Marker	File Name	Size	Modified
<input type="checkbox"/>	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
<input type="checkbox"/>	xonix.wav	30K	02-Oct-05
Delete Marked		Upload File:	

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 the compiler 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:

National Variants	
Marker	Language
<input type="checkbox"/>	arabic
<input type="checkbox"/>	french
<input type="checkbox"/>	russian
<div>Create Language <input type="text"/></div>	

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.

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.

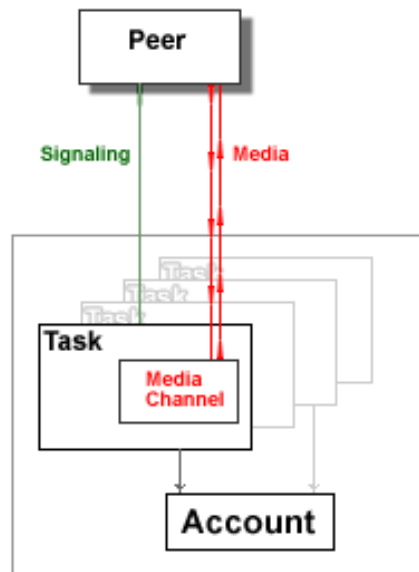
connecting

the Task has initiated an outgoing session (call), but the session has not been established yet.

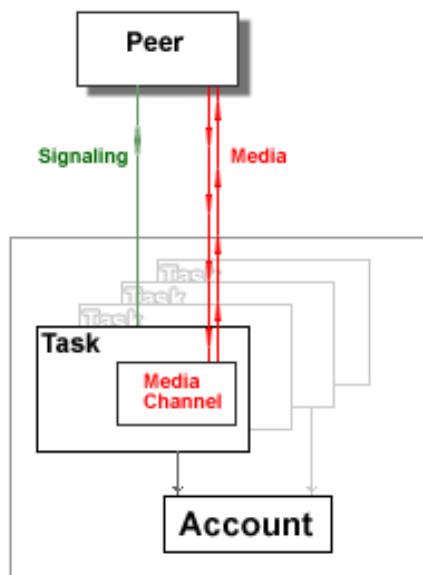
connected

the Task has accepted an incoming session, or the Task has established an outgoing session.

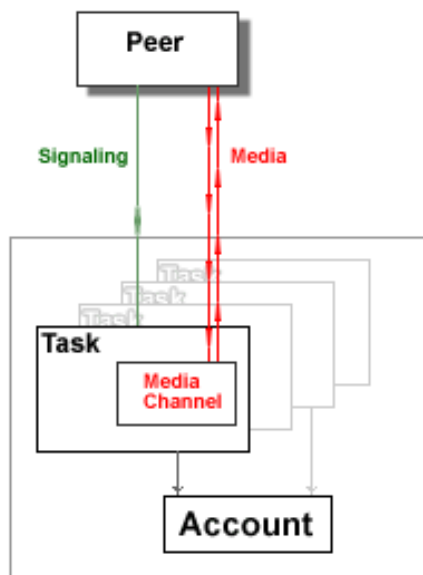
Step 1.



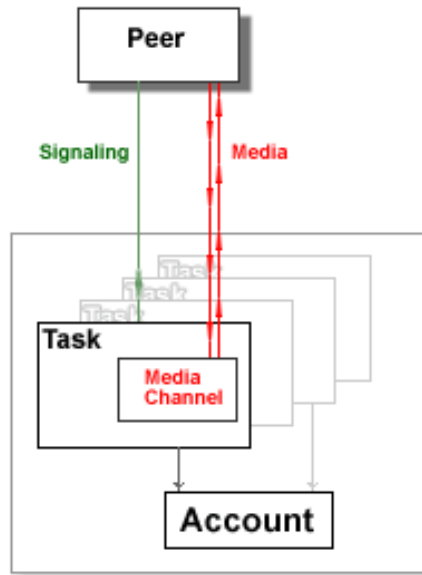
Step 2.



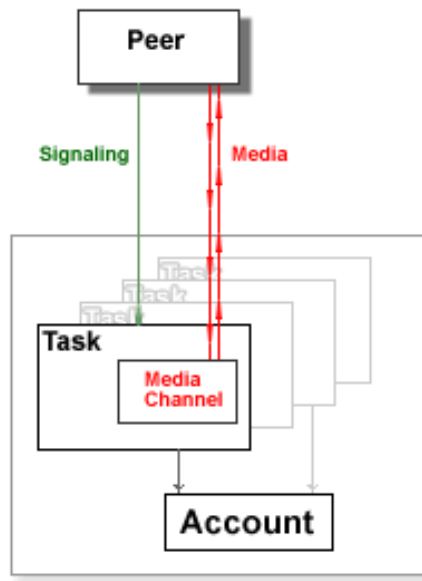
Step 3.



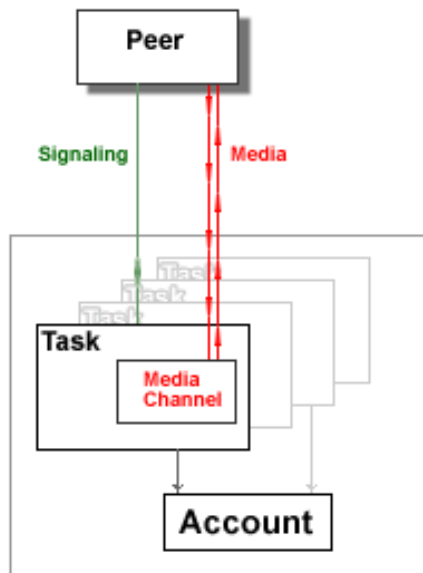
Step 4.



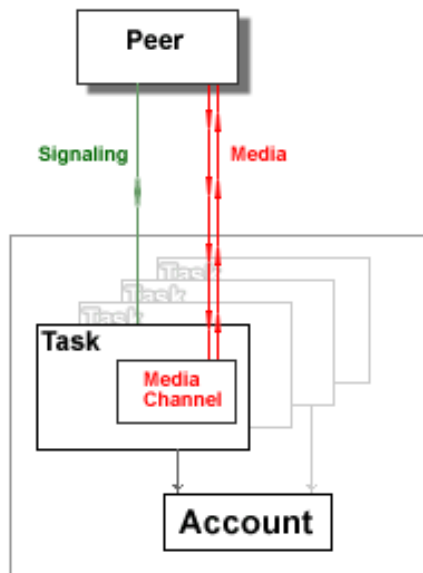
Step 5.



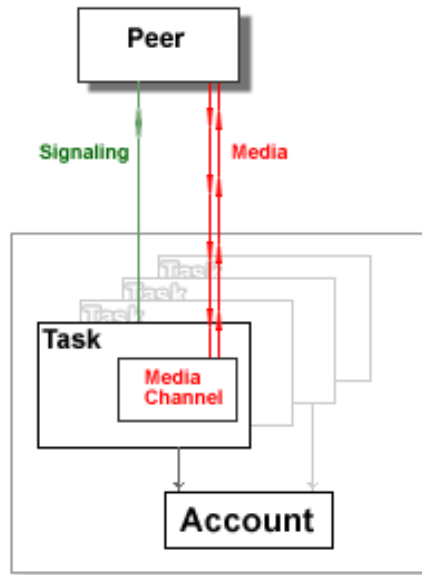
Step 6.



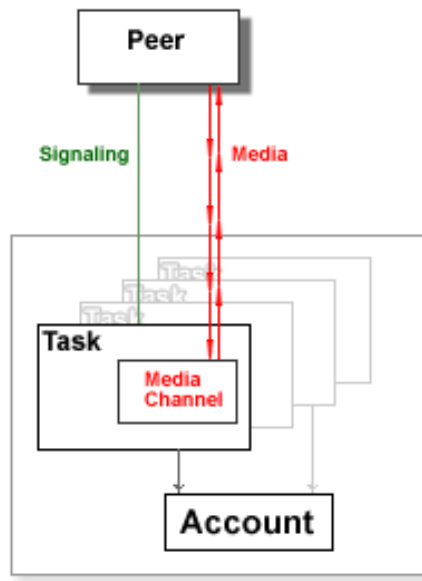
Step 7.



Step 8.



Step 9.



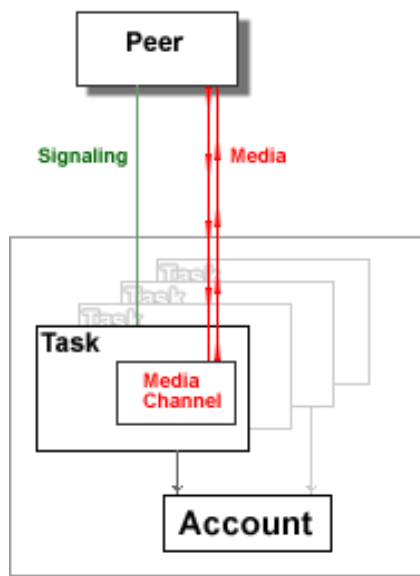
A Task can receive **Signals** from its peer, and it can send Signals itself. Signals can be used to end the current session, to update the session parameters, etc. Some of the Signals sent by the peer are processed by the Real-Time Application Environment itself, while other Signals are passed to the Task for processing.

A Real-Time Application Task has a Media Channel associated with it. When a session is established, the Task Media Channel is connected to the peer's media channel.

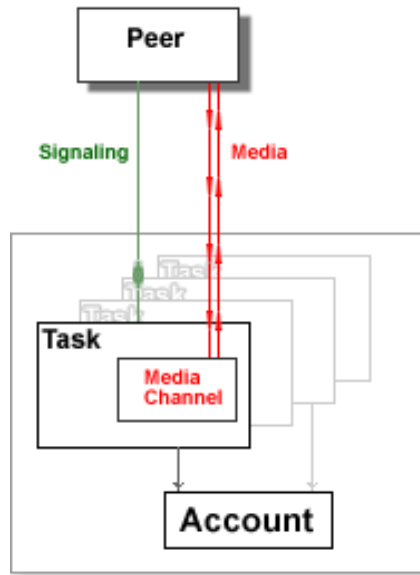
A Task can use its Media Channel to send media (audio, video) to the peer, and to record media sent by the peer.

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.

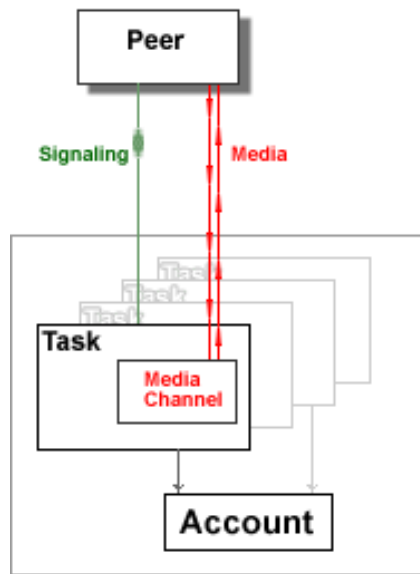
Step 1.



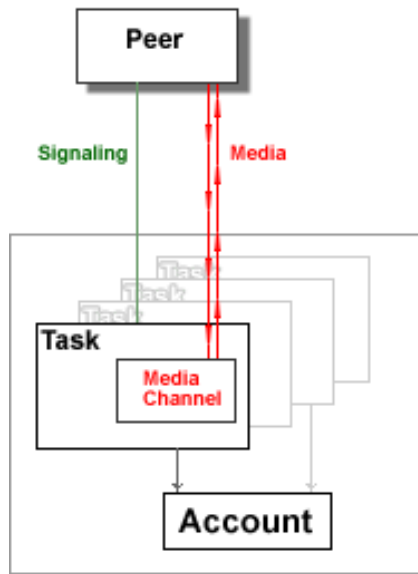
Step 2.



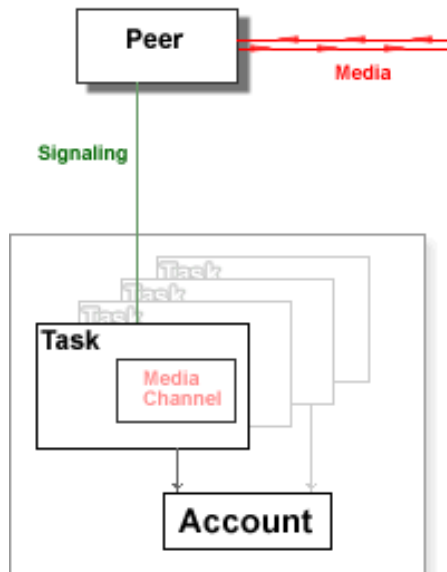
Step 3.



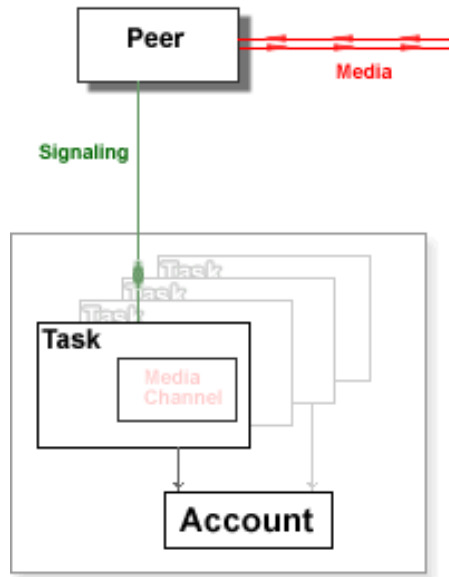
Step 4.



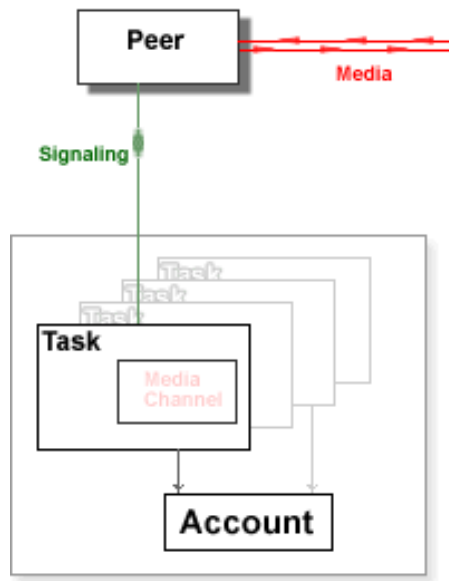
Step 5.



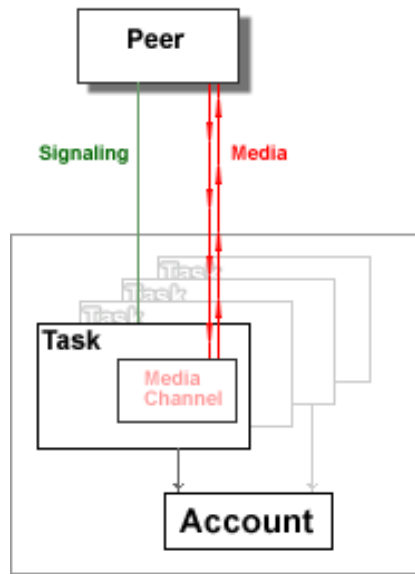
Step 6.



Step 7.

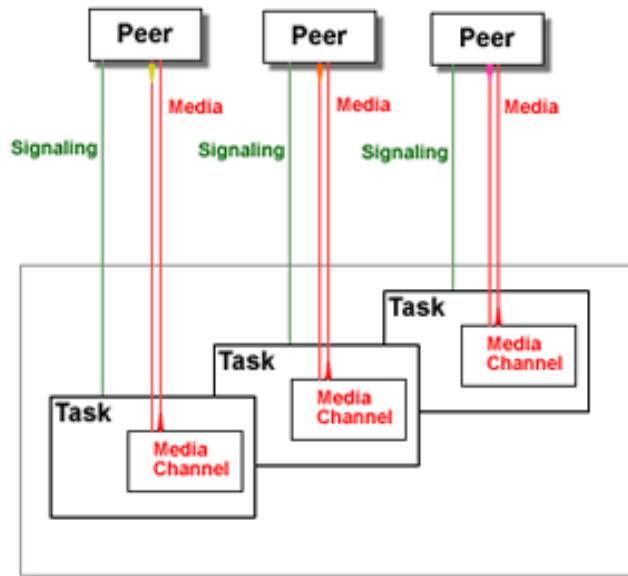


Step 8.

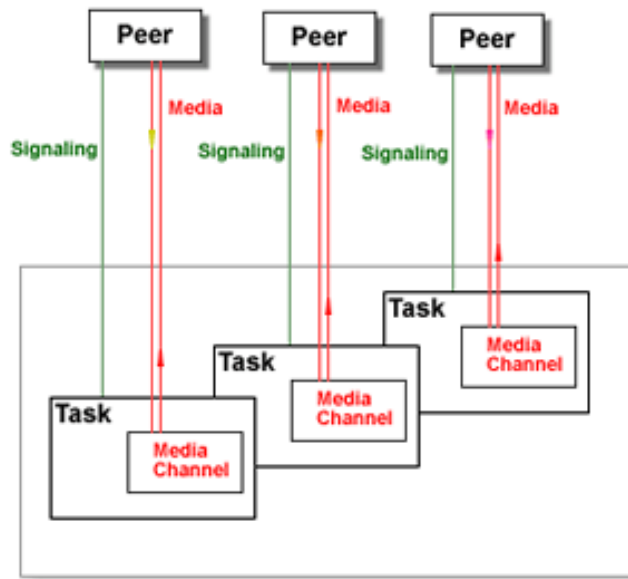


A Media Channel provides a *conversation space*. A Task can attach other peer media to the Task own Media Channel. This operation creates a conversation space (or a *conference*) that includes the Task own peer and all peers with media attached to this Task. Media sent by any peer in a *conversation space* is relayed to all other peers in that space, using the data mixing algorithms and parameters defined for that media type.

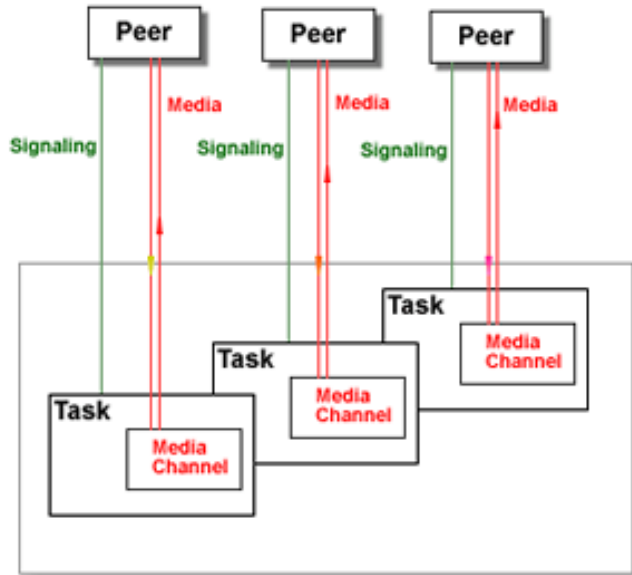
Step 1.



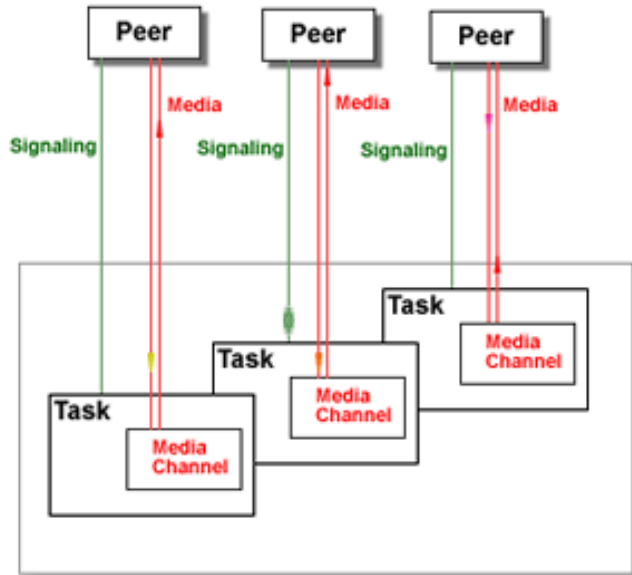
Step 2.



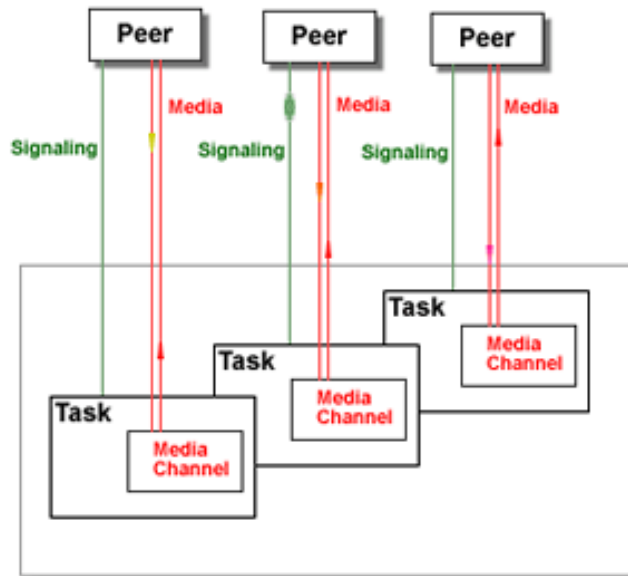
Step 3.



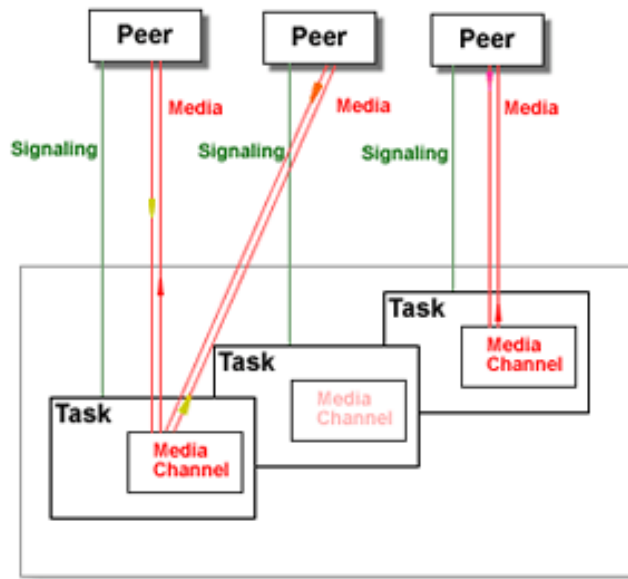
Step 4.



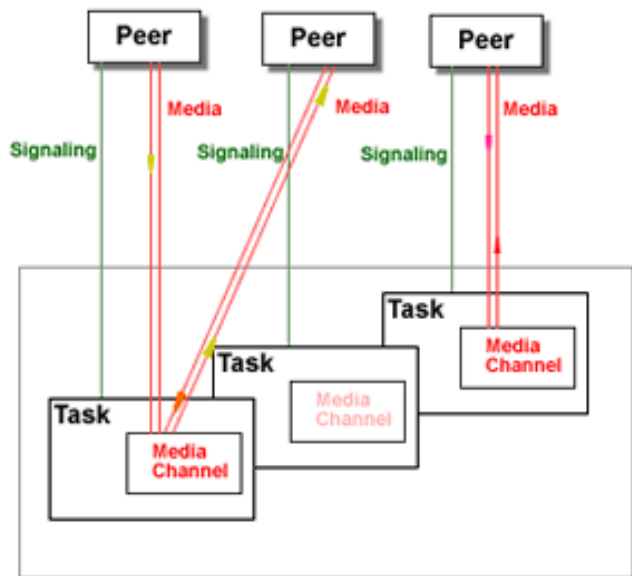
Step 5.



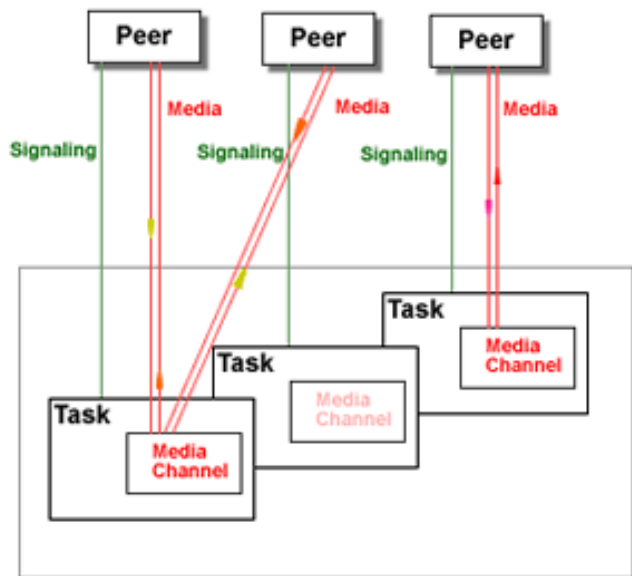
Step 6.



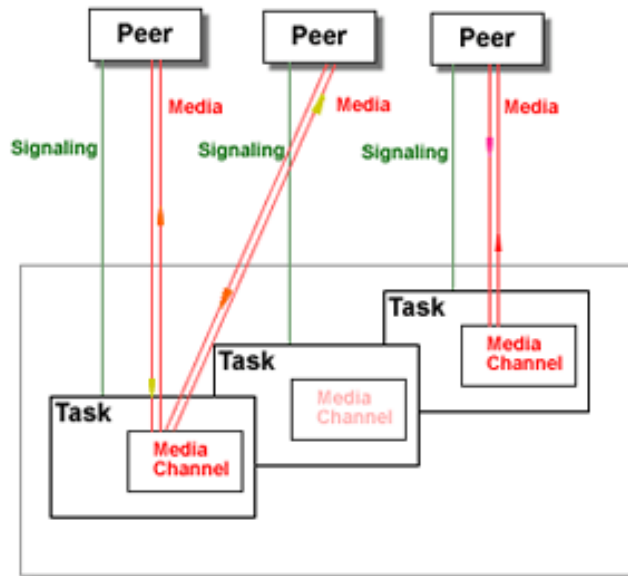
Step 7.



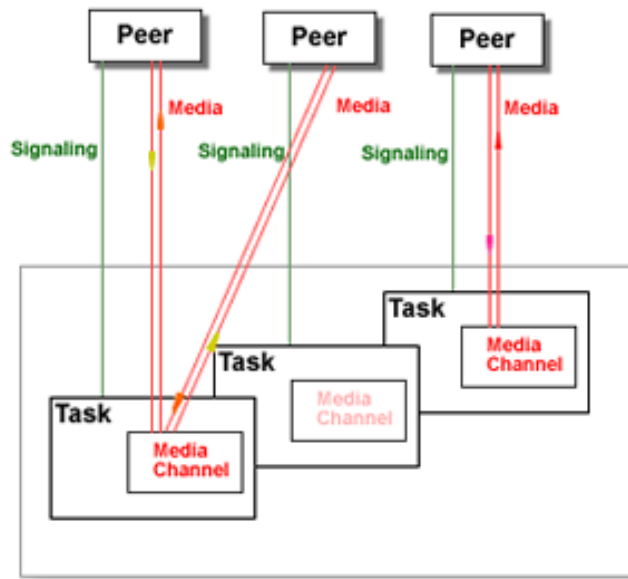
Step 8.



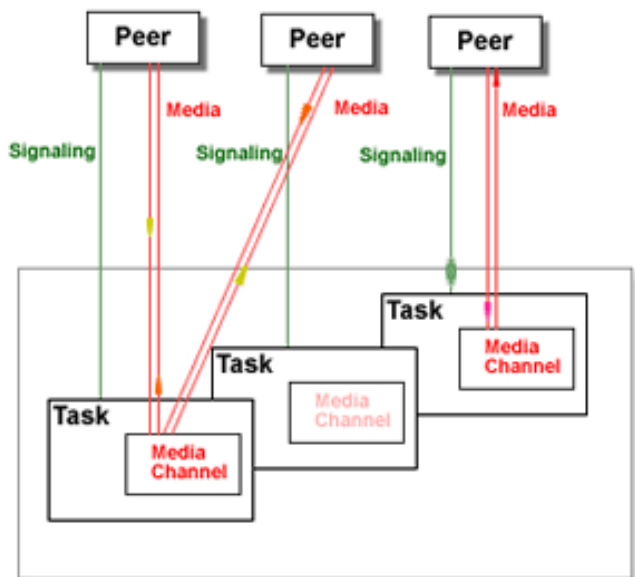
Step 9.



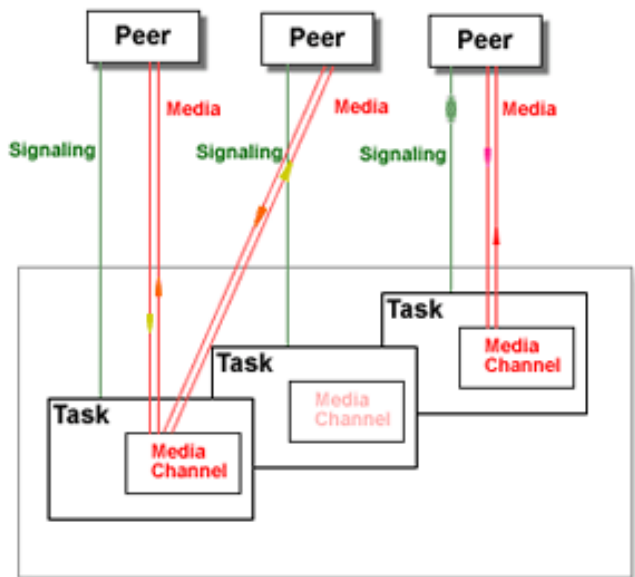
Step 10.



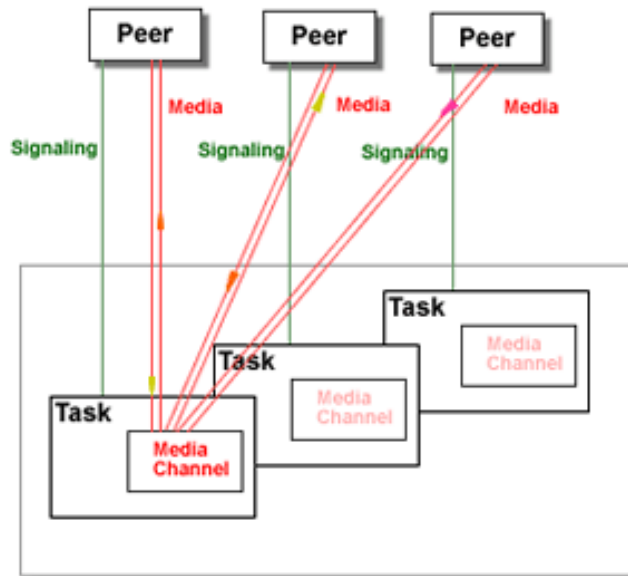
Step 11.



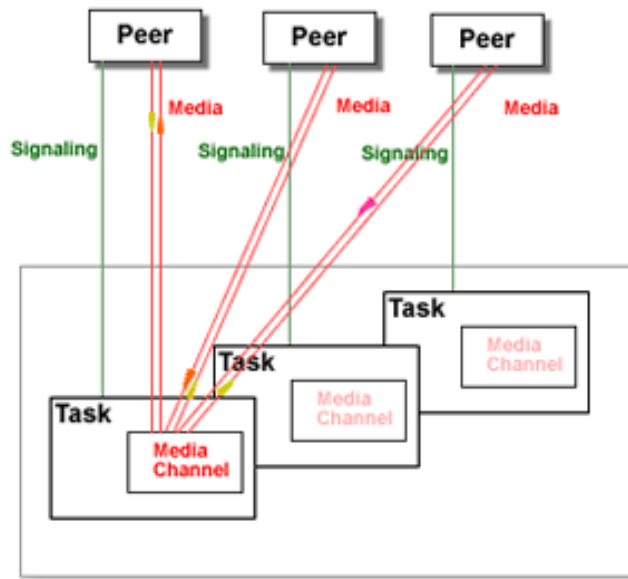
Step 12.



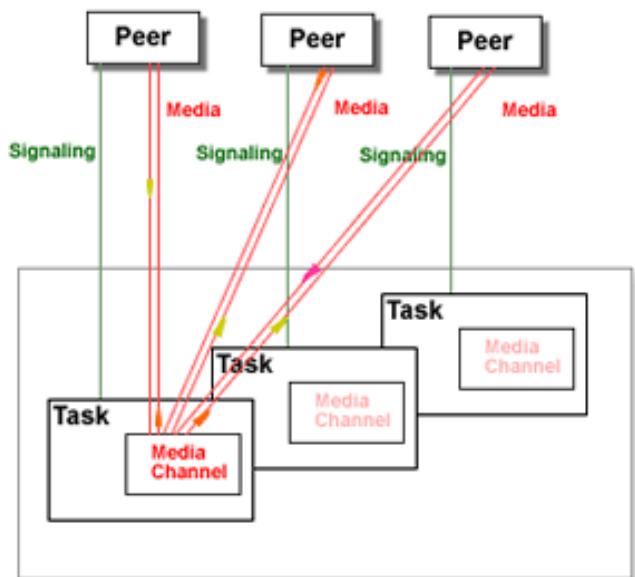
Step 13.



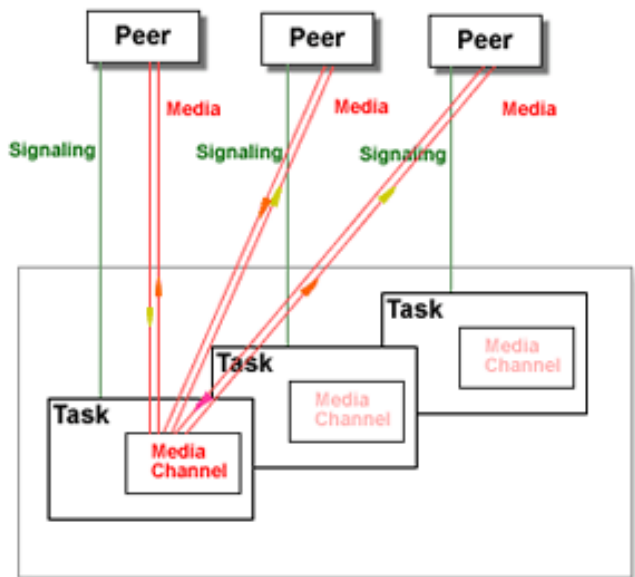
Step 14.



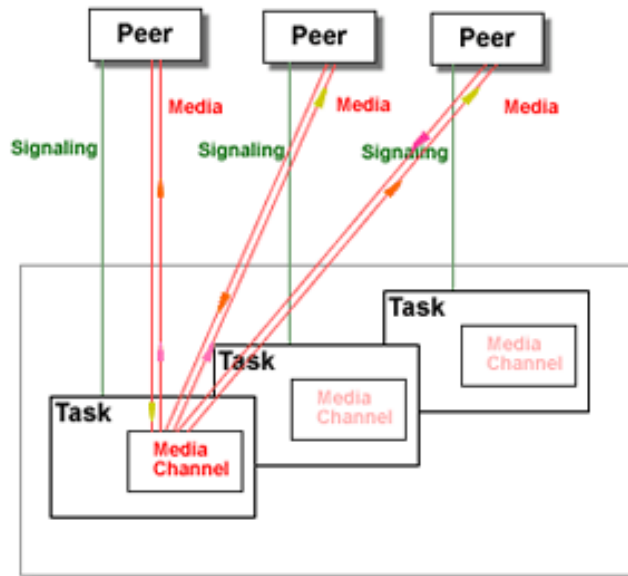
Step 15.



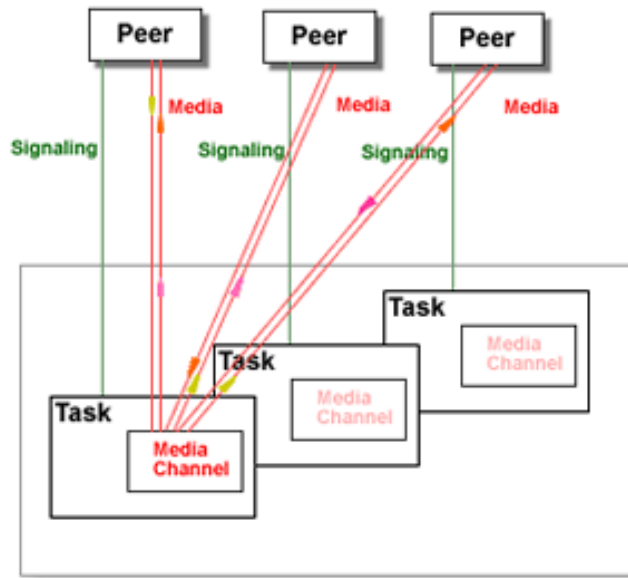
Step 16.



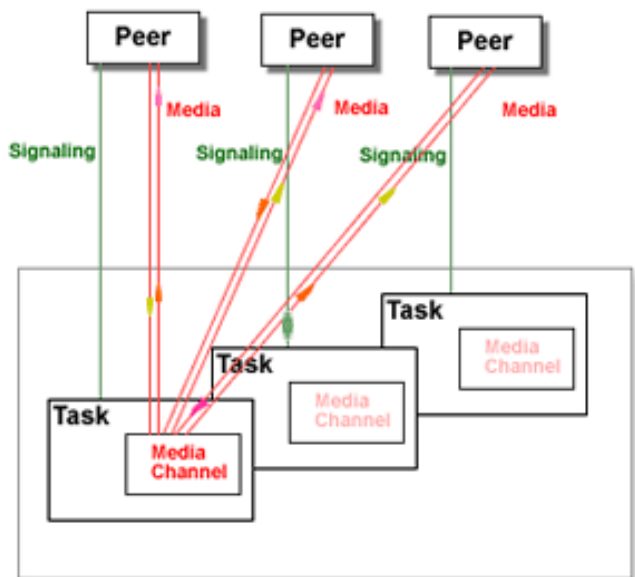
Step 17.



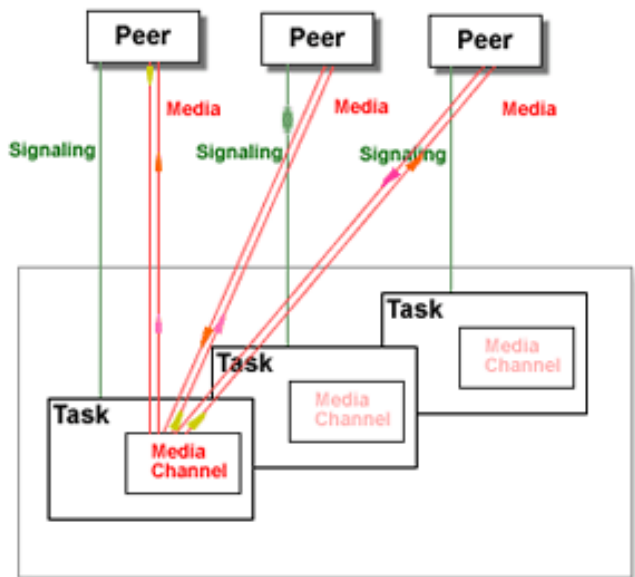
Step 18.



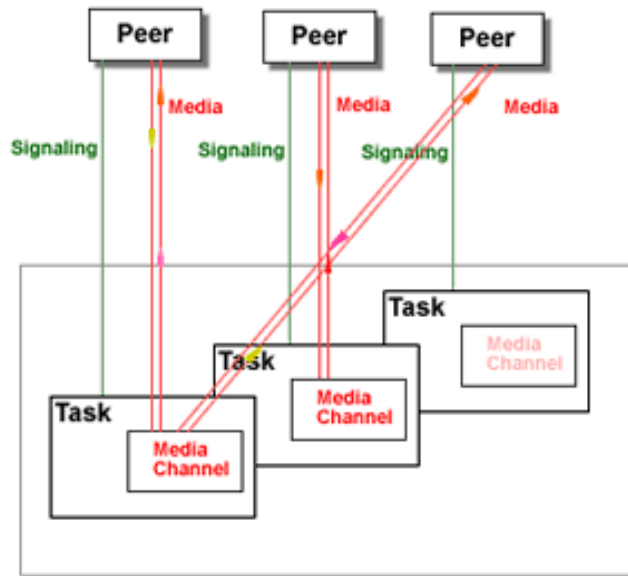
Step 19.



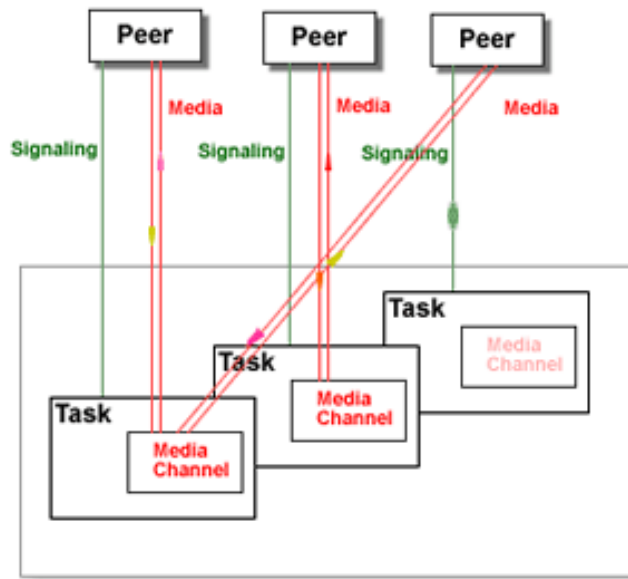
Step 20.



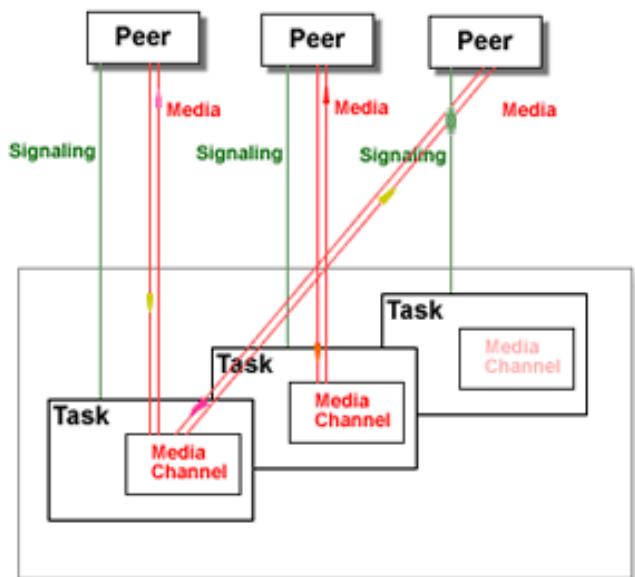
Step 21.



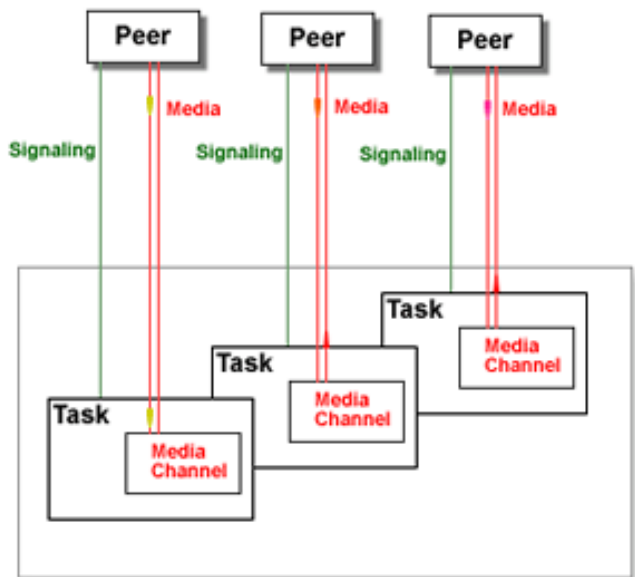
Step 22.



Step 23.



Step 24.



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.

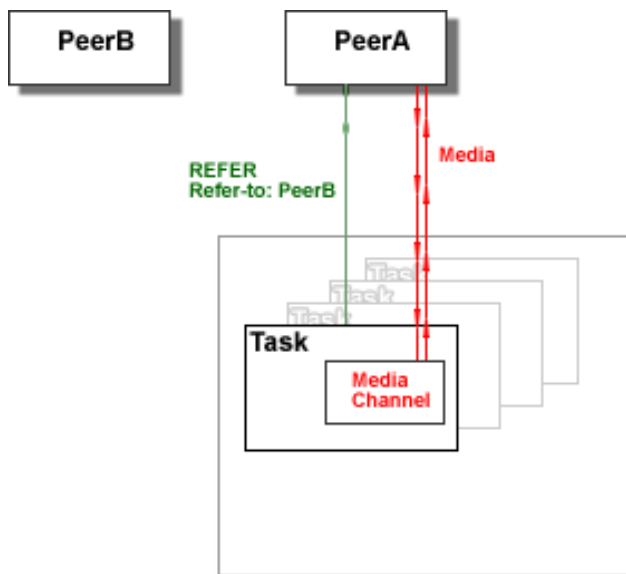
When a Task receives a REFER request (from a remote client, or from a different Task), it can:

- process it automatically (this is the default option)
- reject it
- pass it to the application

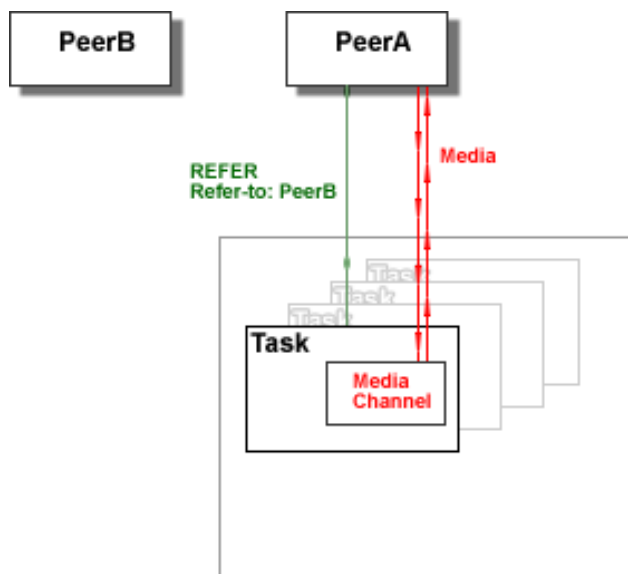
If a Task is instructed to process REFER requests automatically, it creates a new INVITE request and sends it to the address specified in the Refer-To: field. While this INVITE request is in progress, the task informs the peer about the call progress by sending it NOTIFY requests.

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:

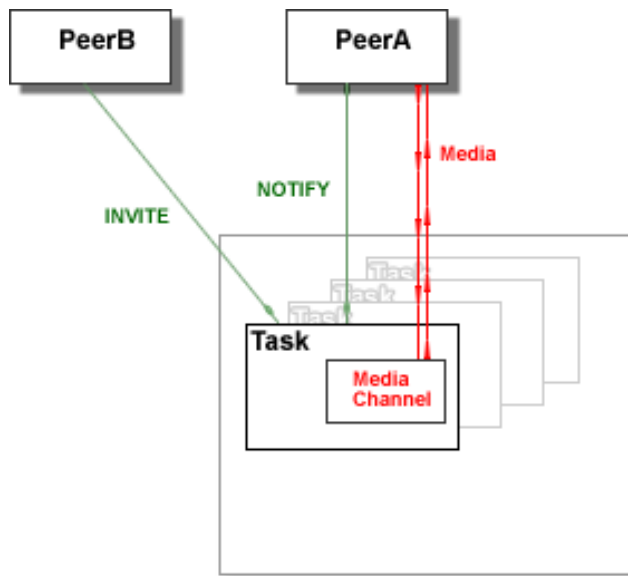
Step 1.



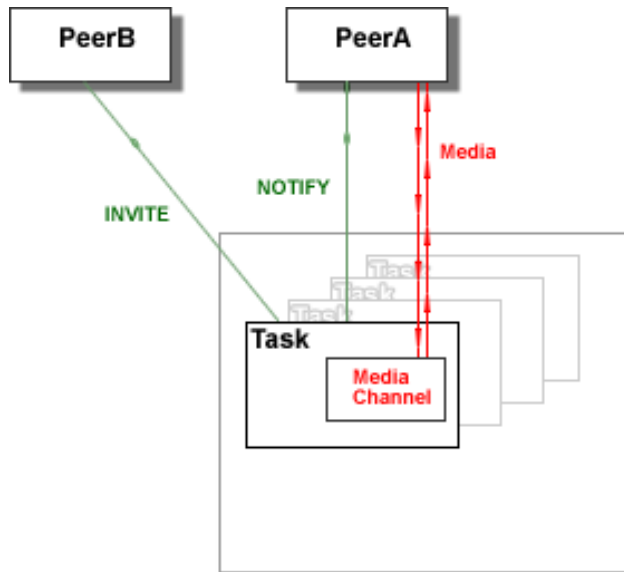
Step 2.



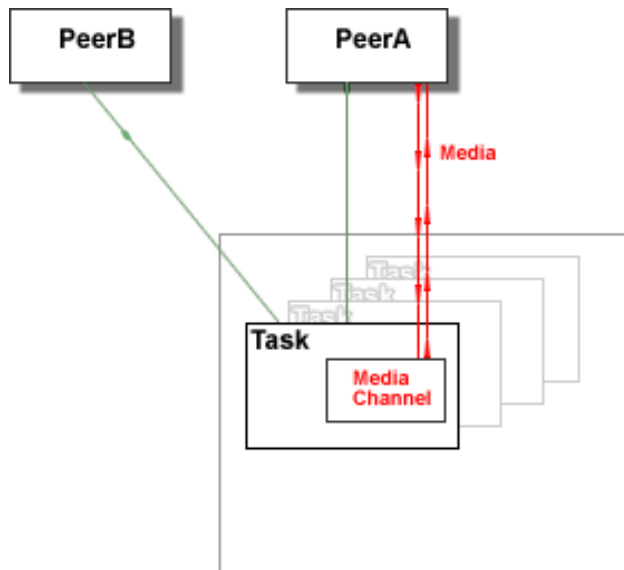
Step 3.



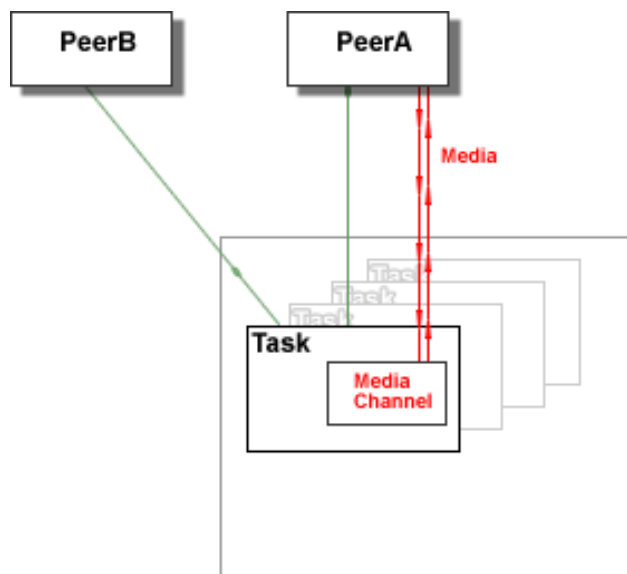
Step 4.



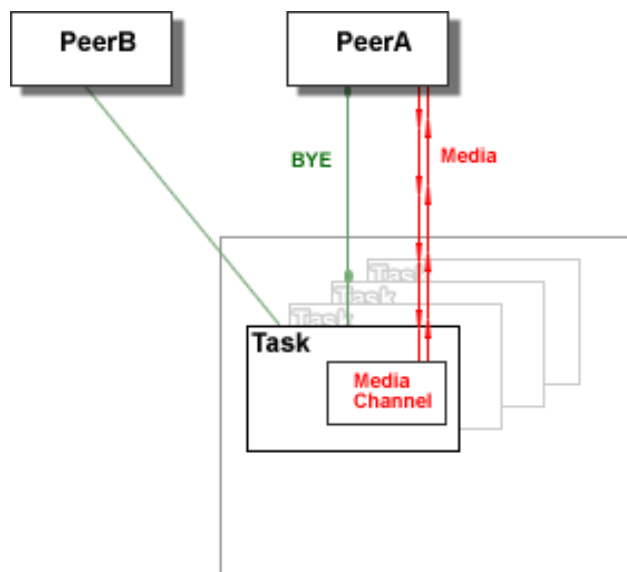
Step 5.



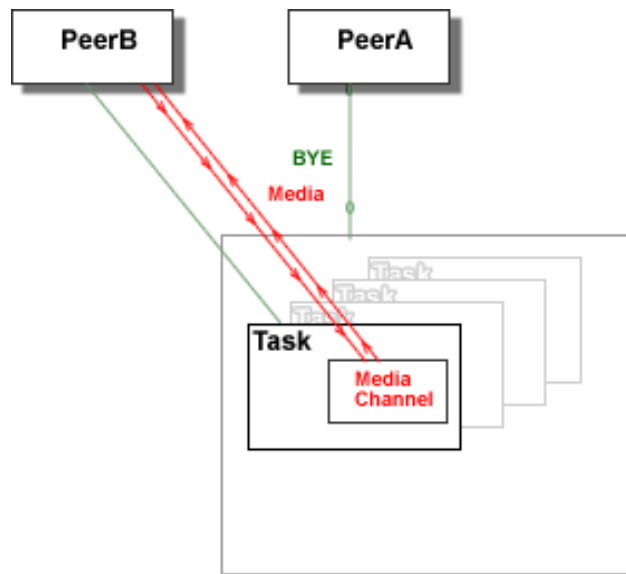
Step 6.



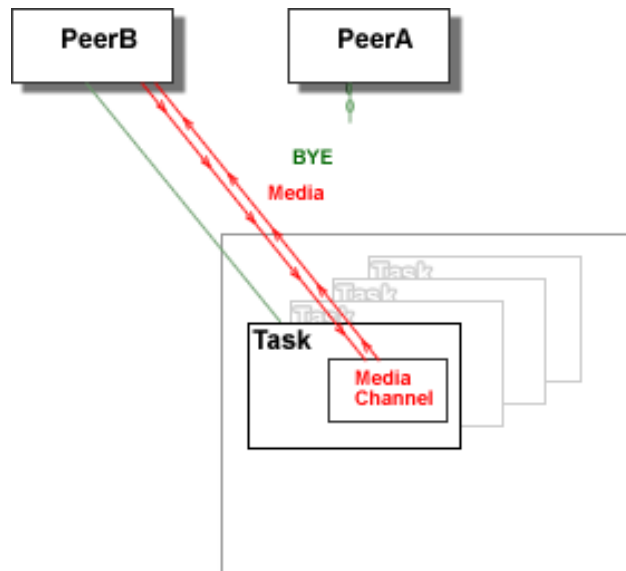
Step 7.



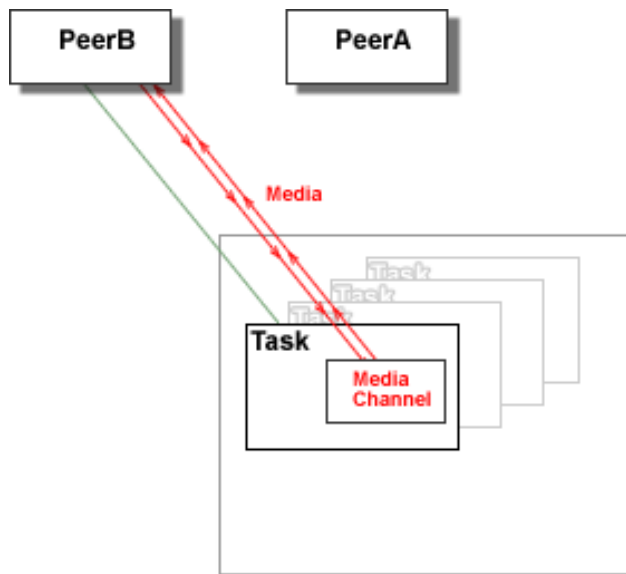
Step 8.



Step 9.



Step 10.



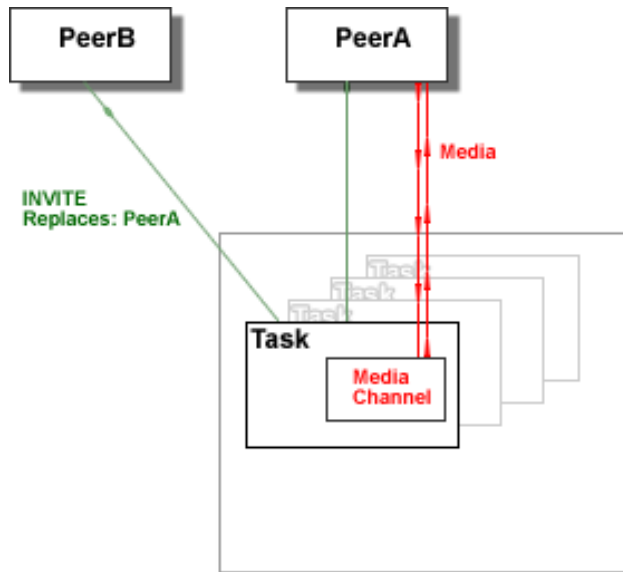
The INVITE signal initiated with the REFER signal can contain authentication information. The application program can instruct the Task:

- to authenticate the INVITE signal as coming from the Account the started the Task.
- to authenticate the INVITE signal as coming from the peer (the entity that has sent the REFER signal). This is the default option.
- to send the INVITE signal without authentication.

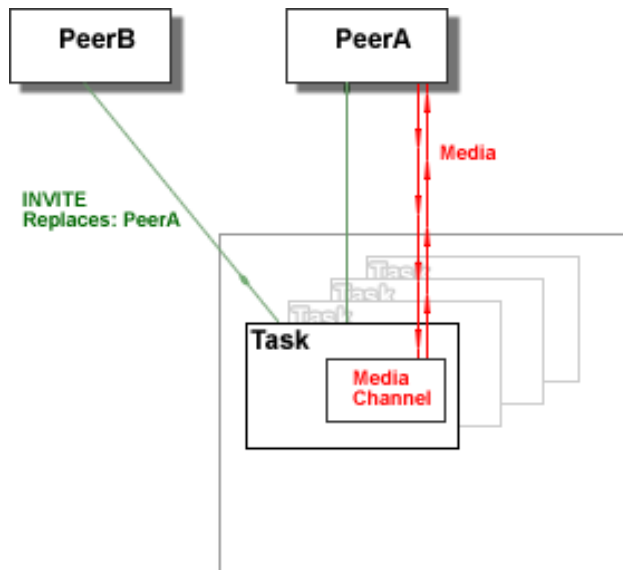
To process "Assisted" Call Transfer, the Real-Time Application engine detects all INVITE signals with the Replaces field. If the dialog data specified in the Replaces field matches an active dialog established in any Real-Time Task, the INVITE signal is directed to that Task. The Task sends a BYE signal 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.

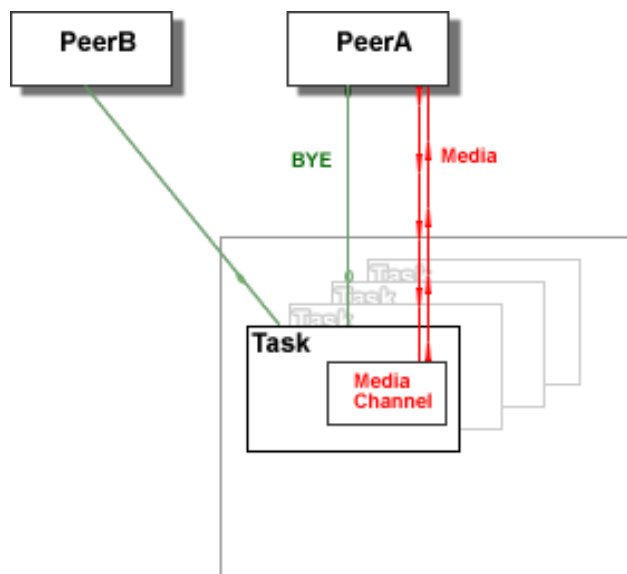
Step 1.



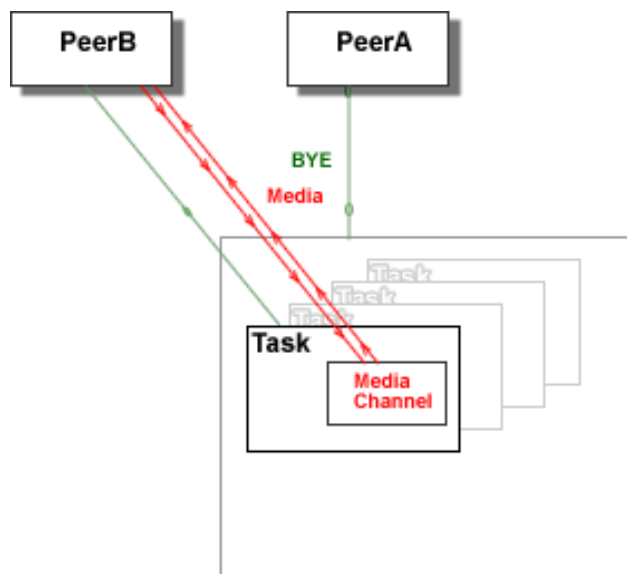
Step 2.



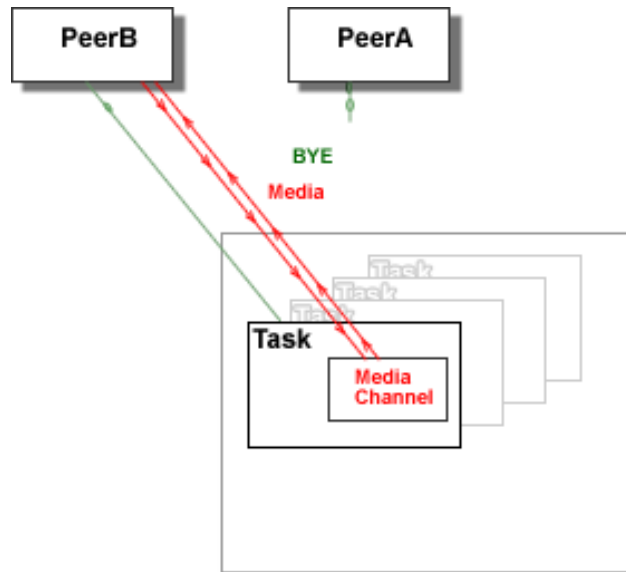
Step 3.



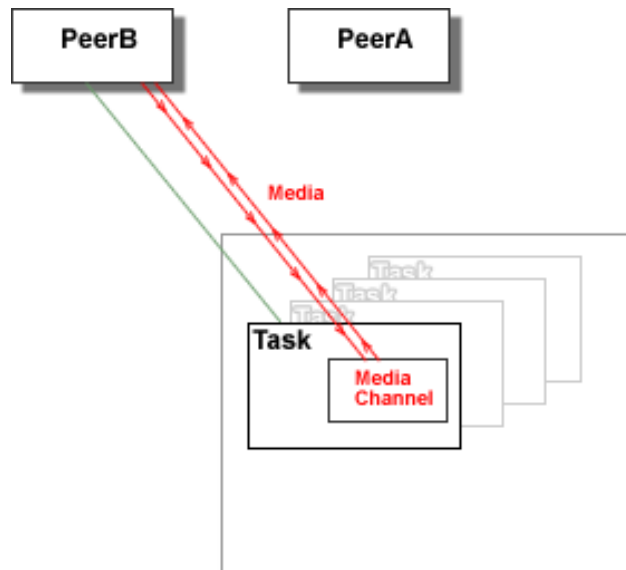
Step 4.



Step 5.



Step 6.



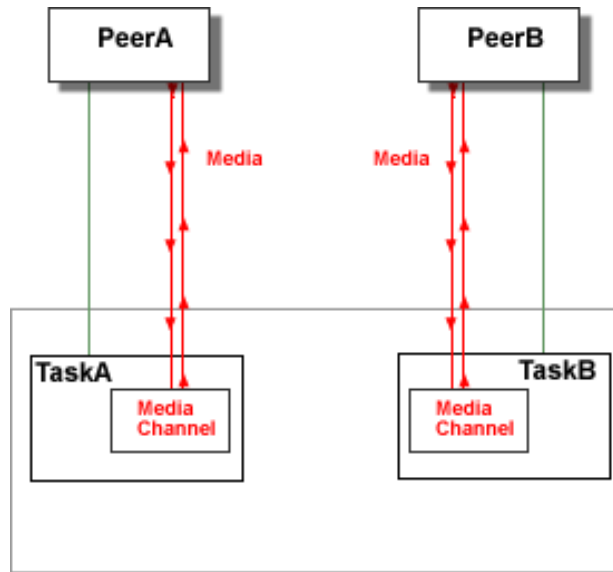
Bridged Calls

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 signalling.

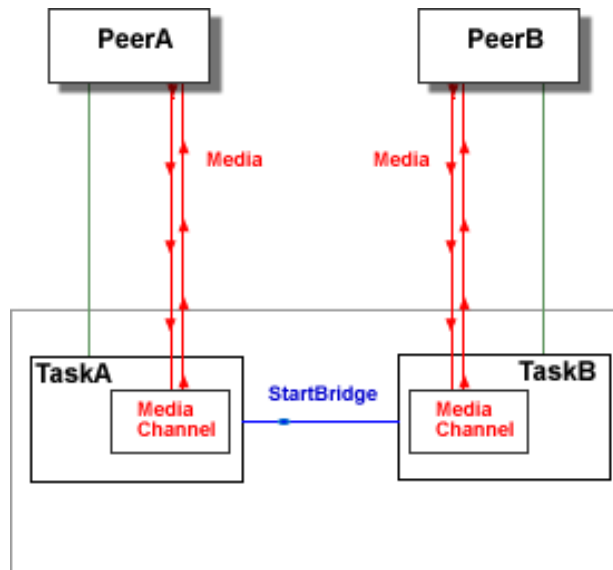
To build a bridge:

- 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.

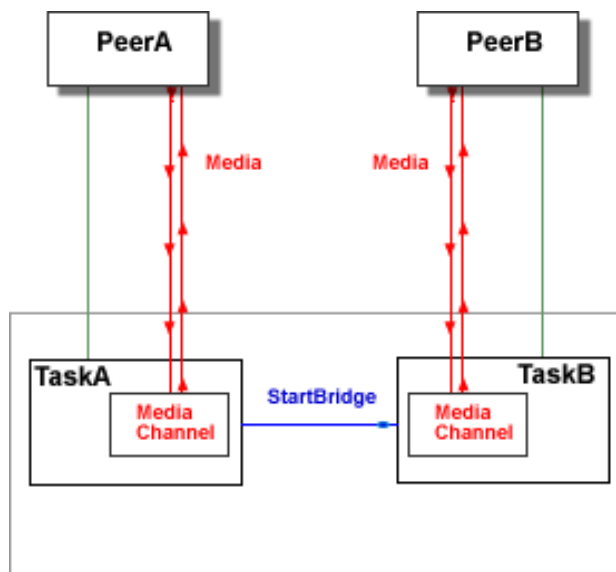
Step 1.



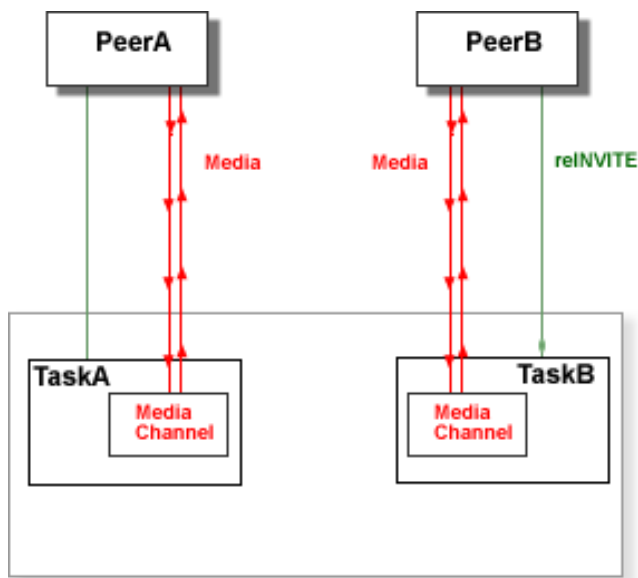
Ste 2.



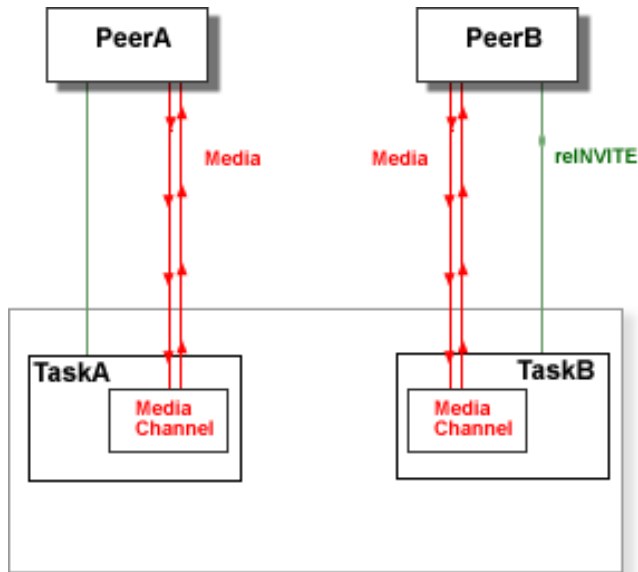
Step 3.



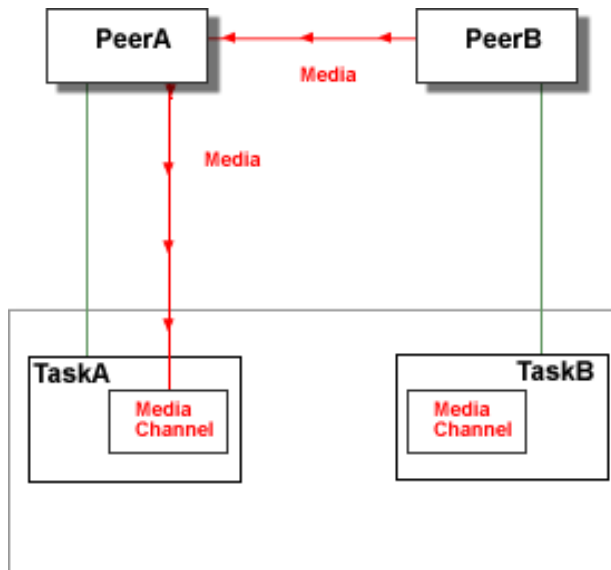
Step 4.



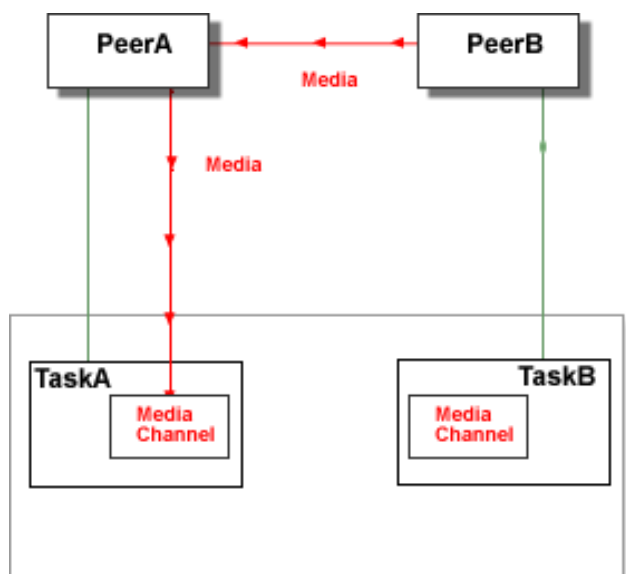
Step 5.



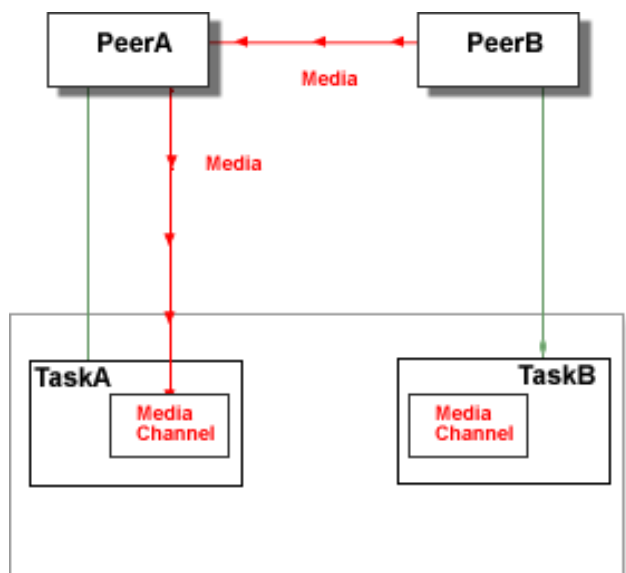
Step 6.



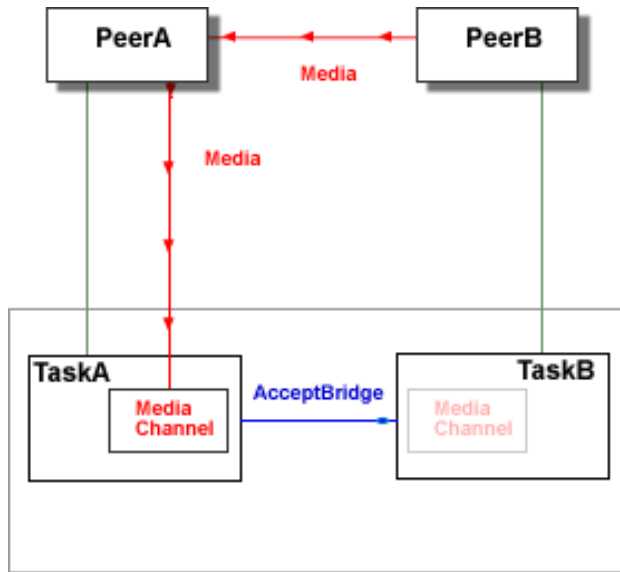
Step 7.



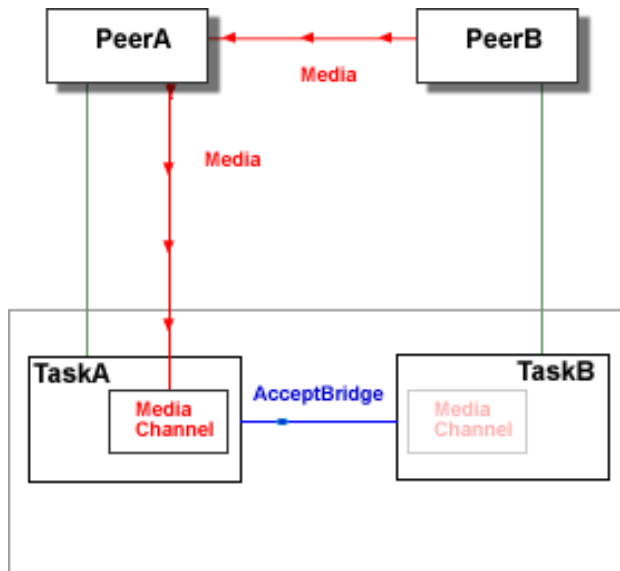
Step 8.



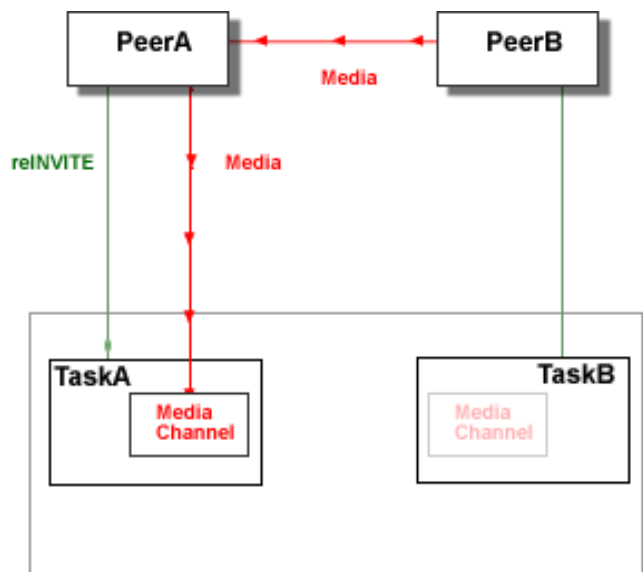
Step 9.



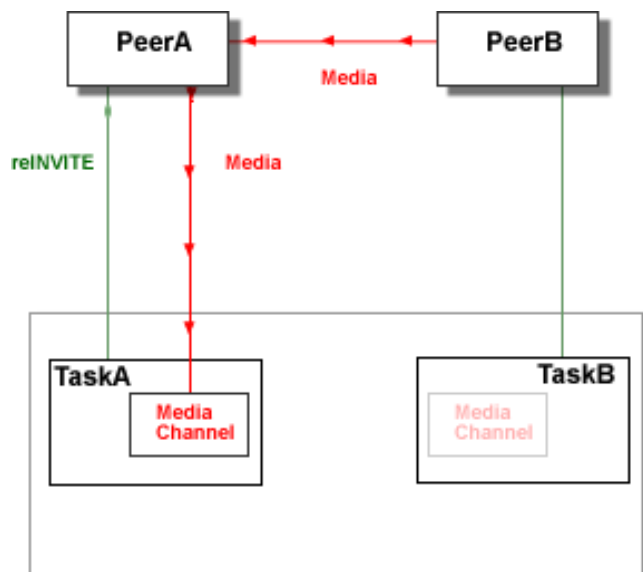
Step 10.



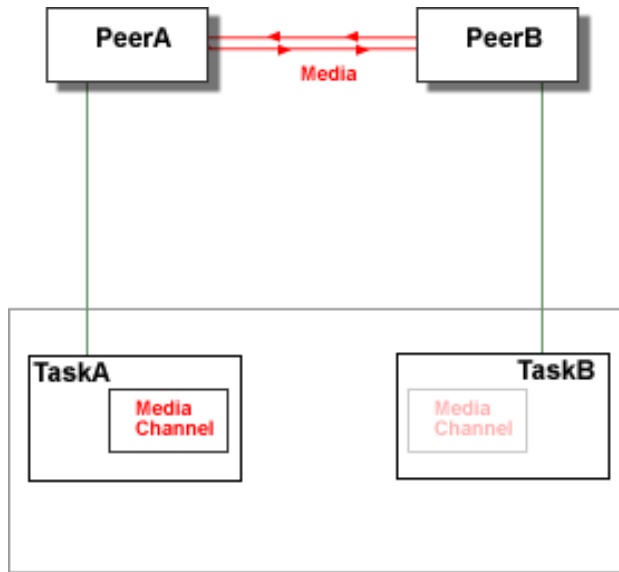
Step 11.



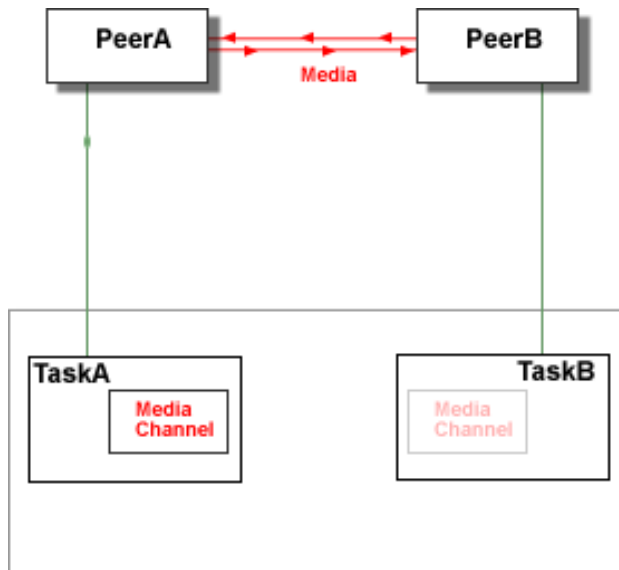
Step 12.



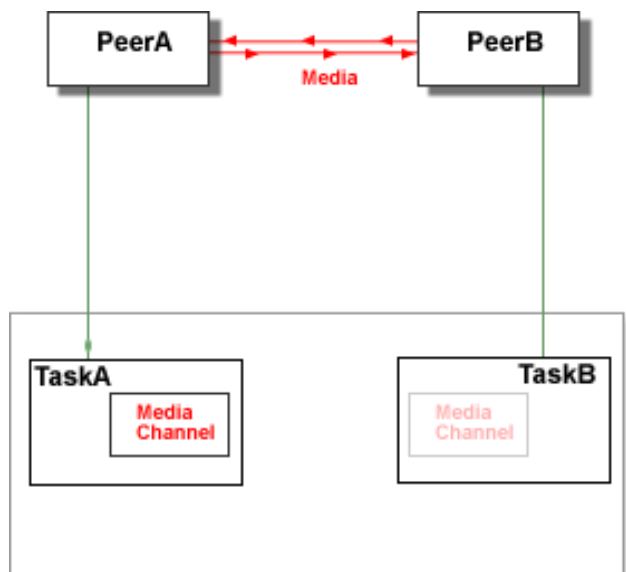
Step 13.



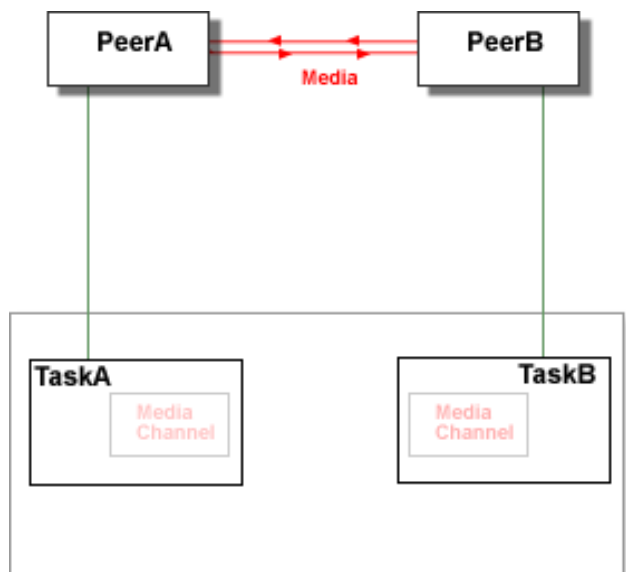
Step 14.



Step 15.



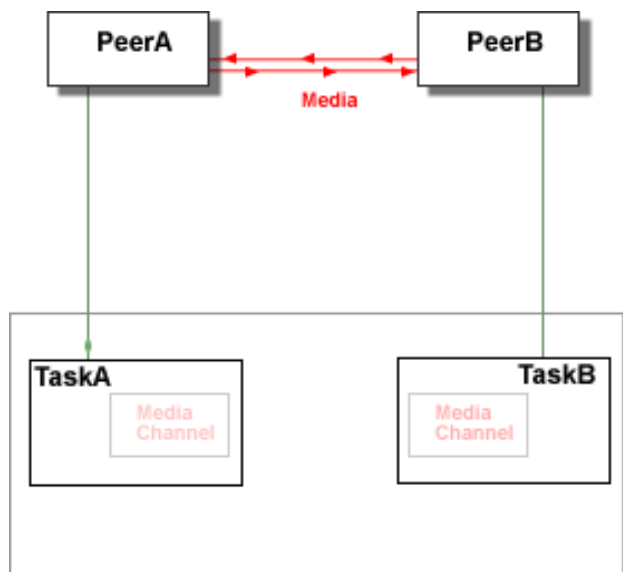
Step 16.



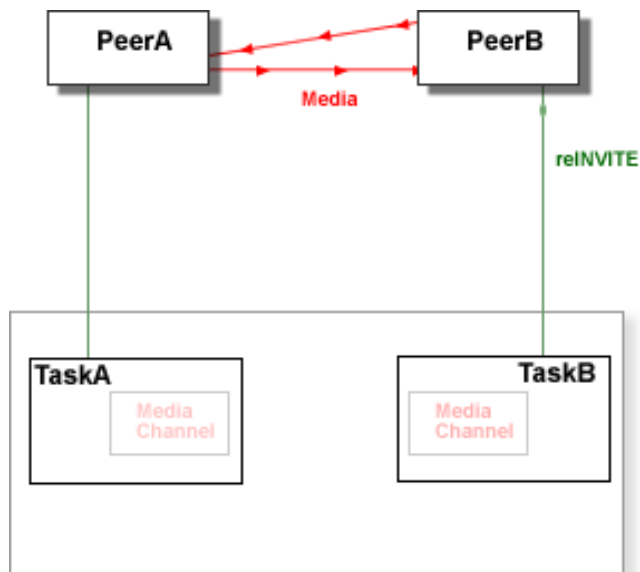
When the Tasks A and B are in the "bridged" state, one of the Tasks (Task B here) can receive a re-INVITE signal from its peer. The new media description in this signal request may ask the Task to change its media parameters (hold/resume, switching to a different media source, etc.). This re-INVITE signal 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 processes 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.

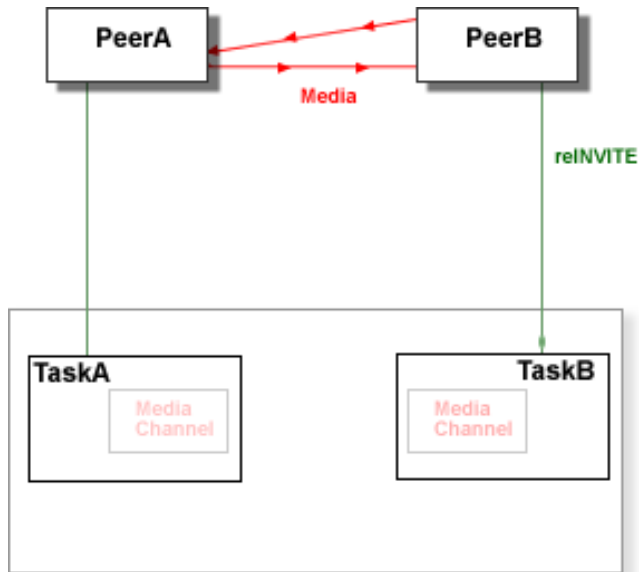
Step 1.



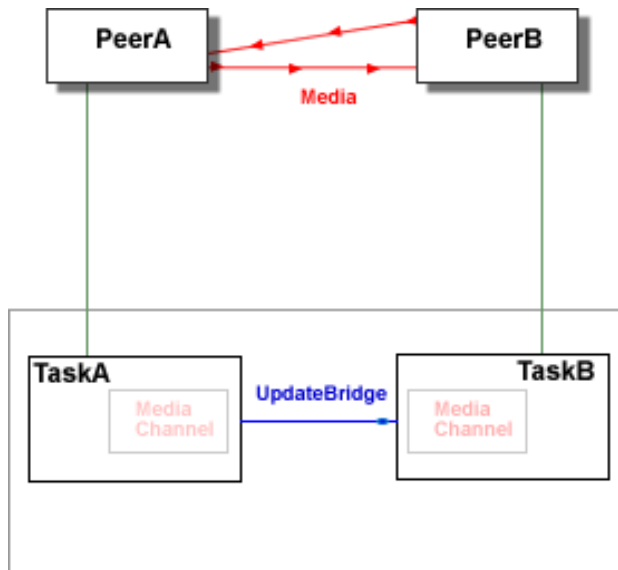
Step 2.



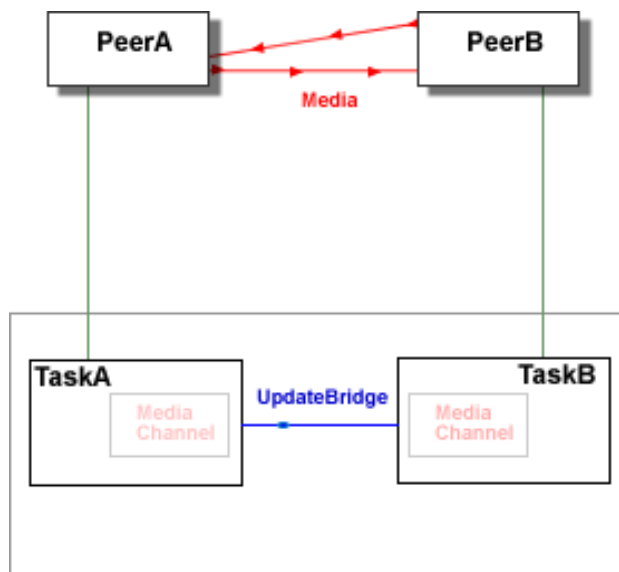
Step 3.



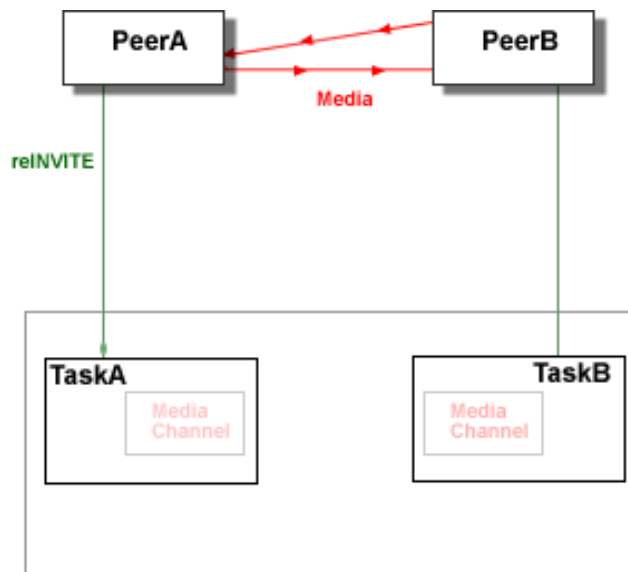
Step 4.



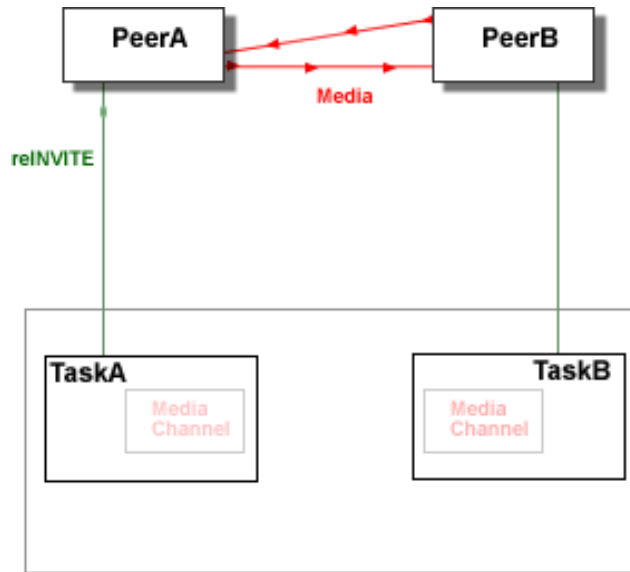
Step 5.



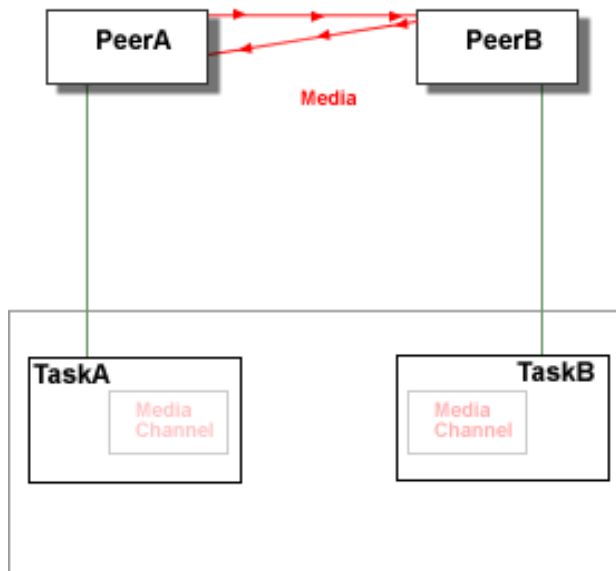
Step 6.



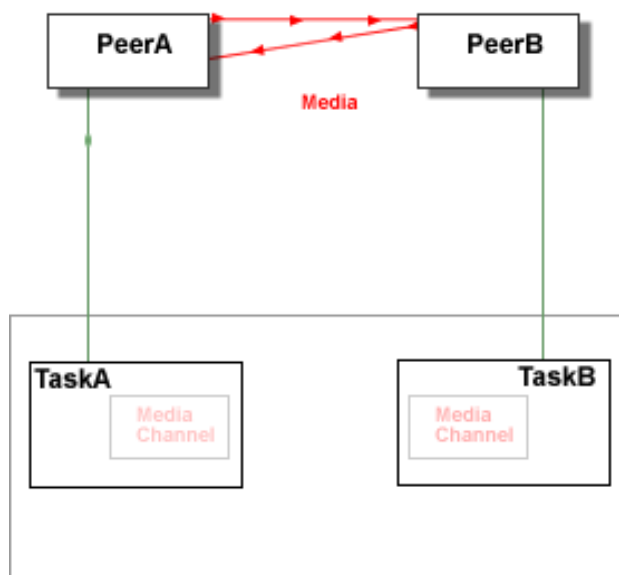
Step 7.



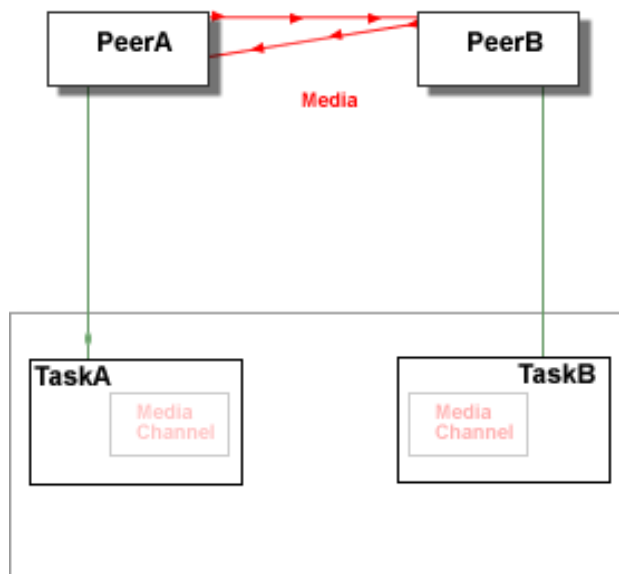
Step 8.



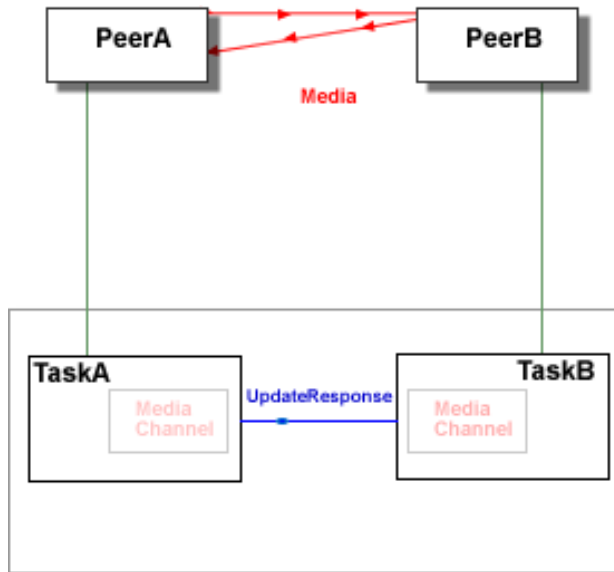
Step 9.



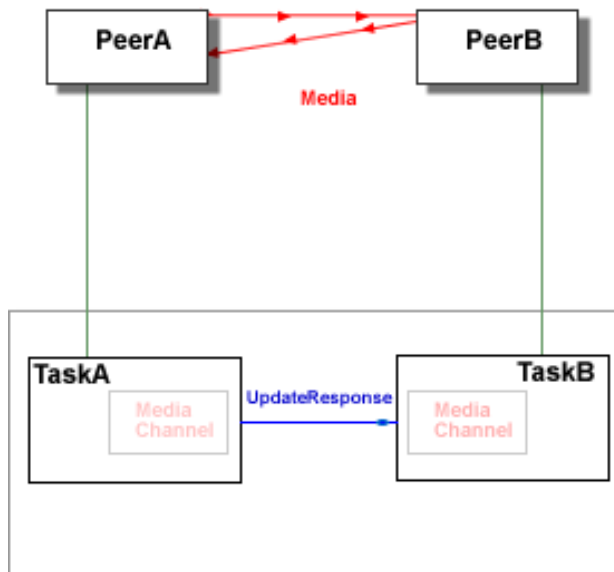
Step 10.



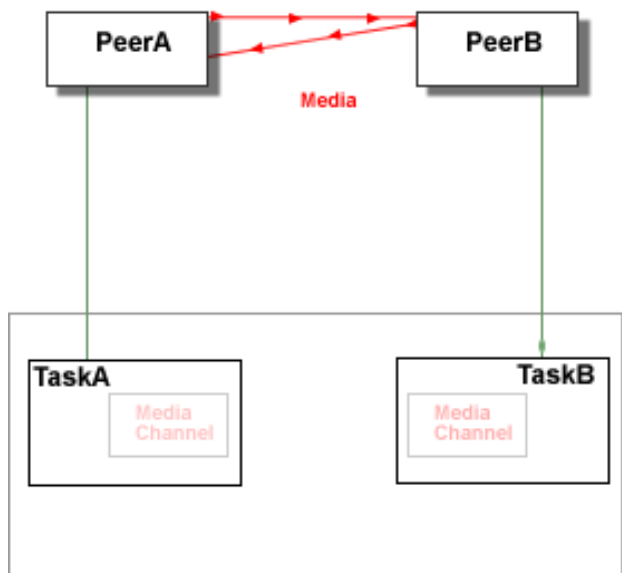
Step 11.



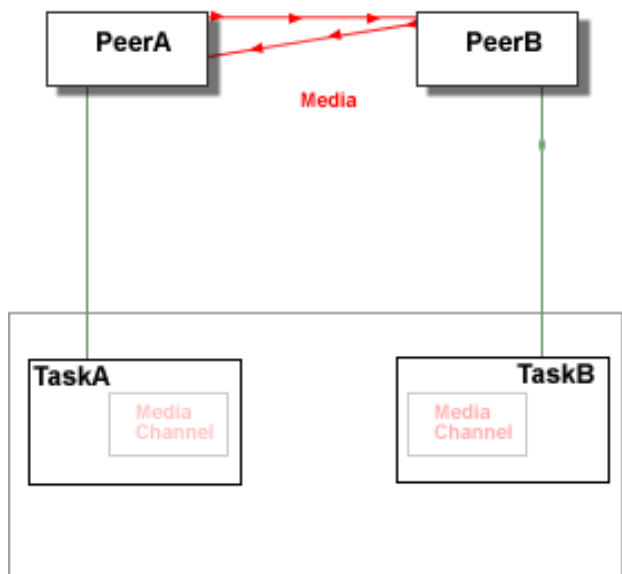
Step 12.



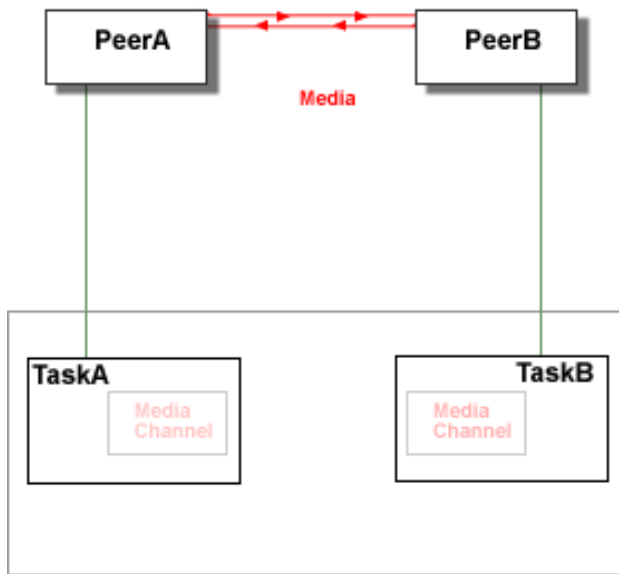
Step 13.



Step 14.



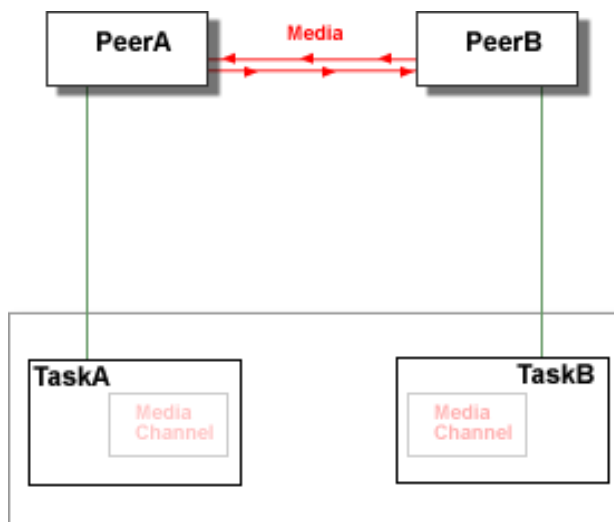
Step 15.



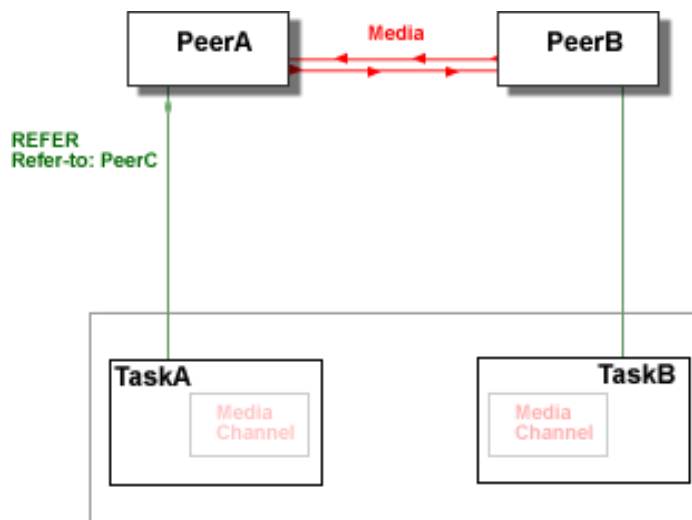
When the Tasks A and B are in the "bridged" state, one of the Tasks (Task A here) can receive a Call Transfer (REFER) signal from its peer:

- 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.

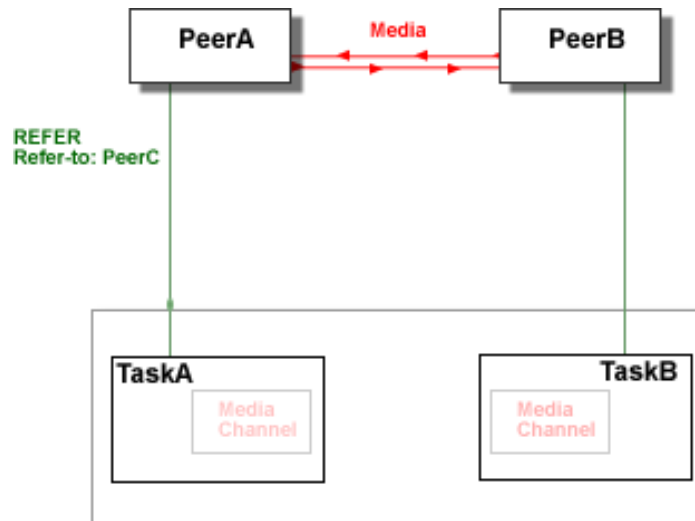
Step 1.



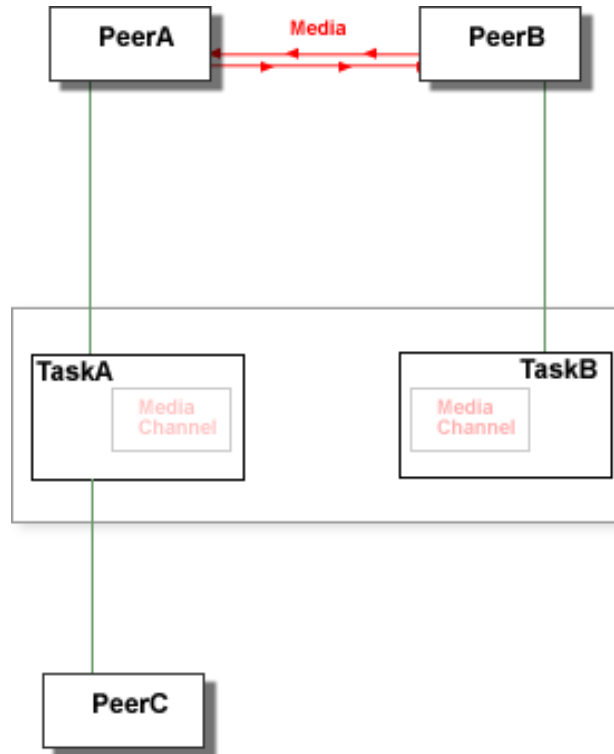
Step 2.



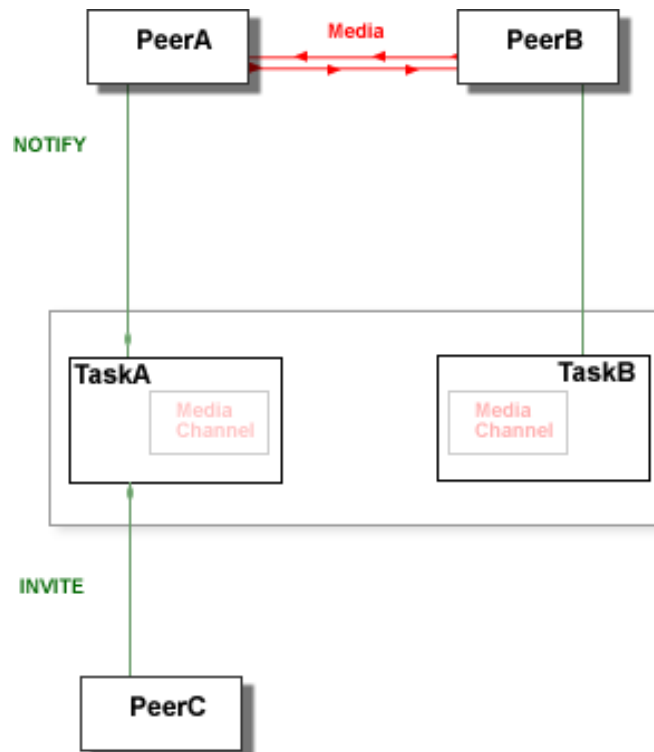
Step 3.



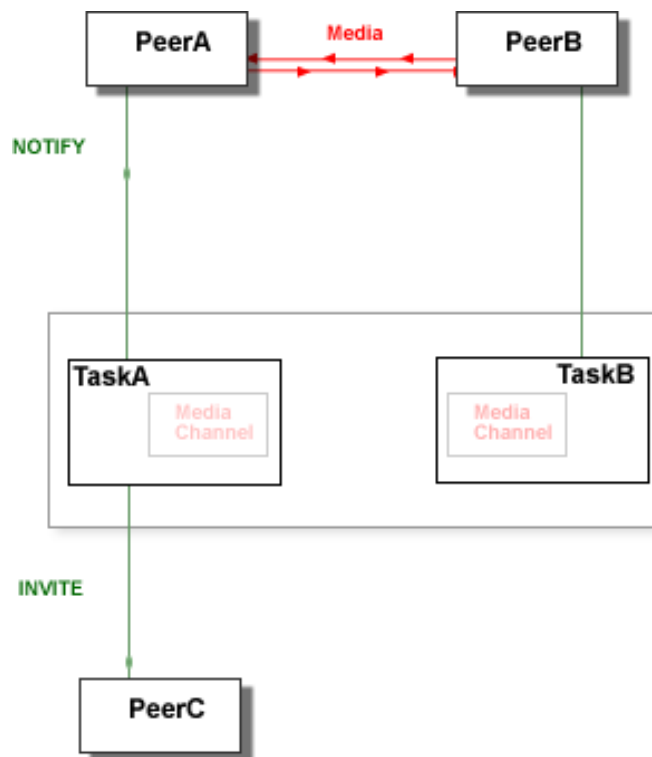
Step 4.



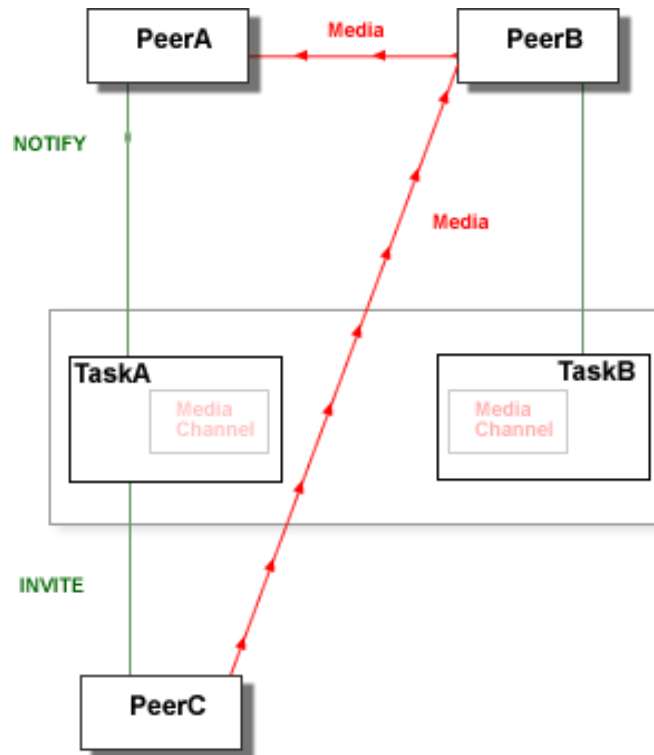
Step 5.



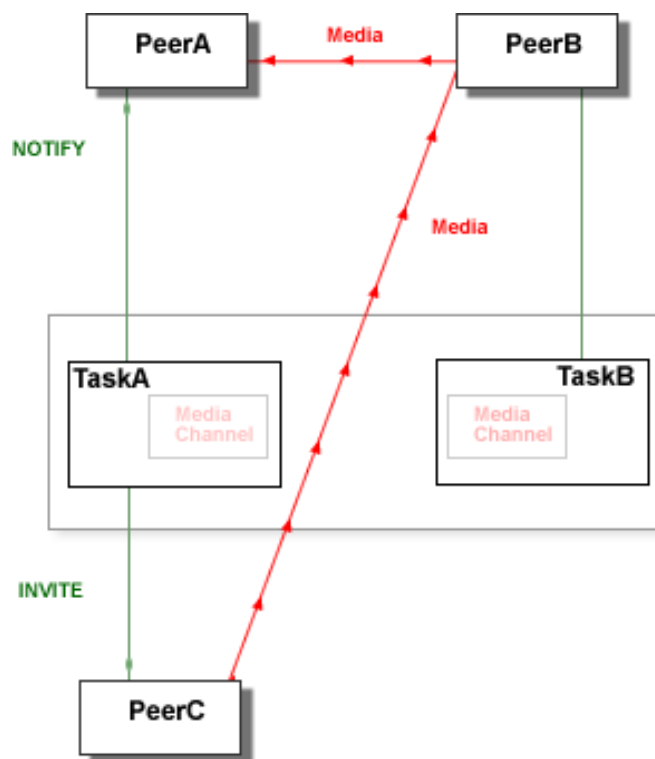
Step 6.



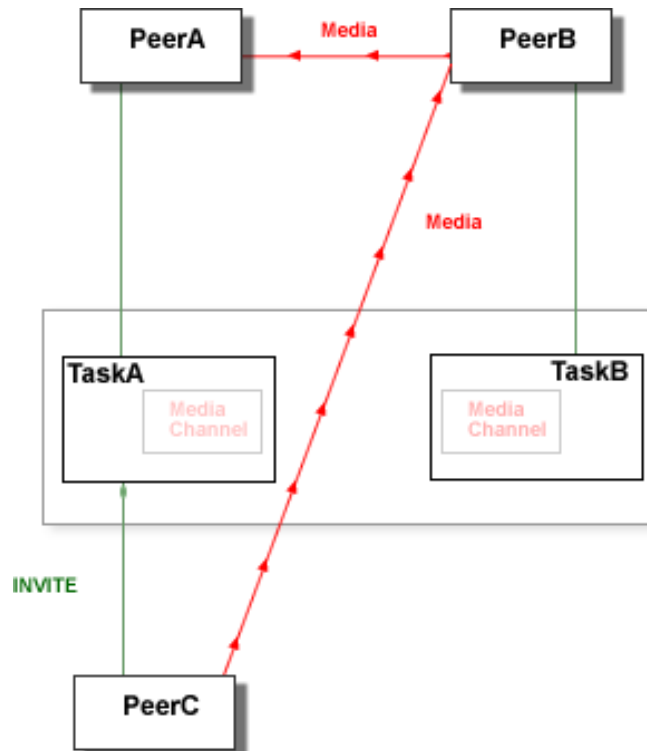
Step 7.



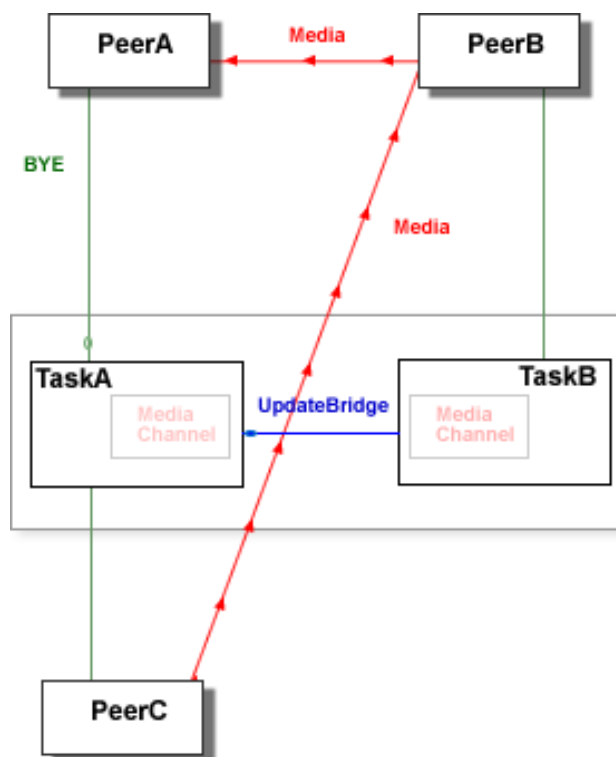
Step 8.



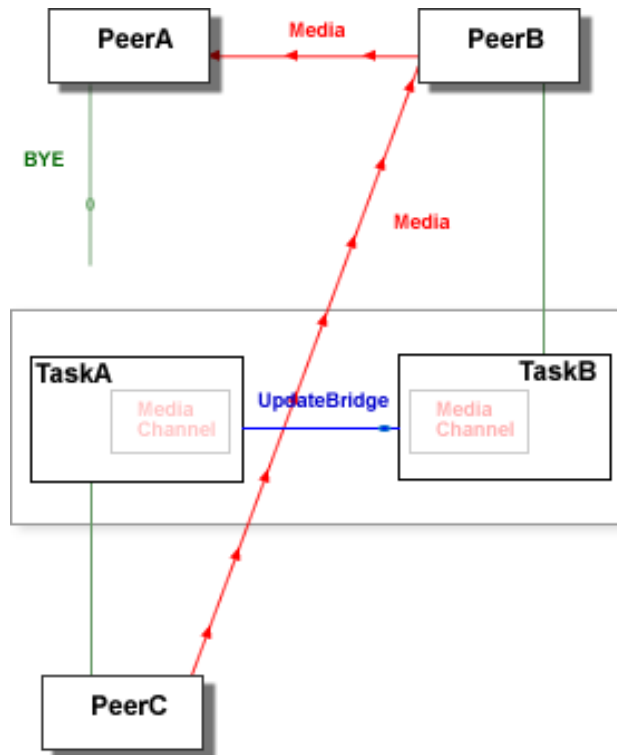
Step 9.



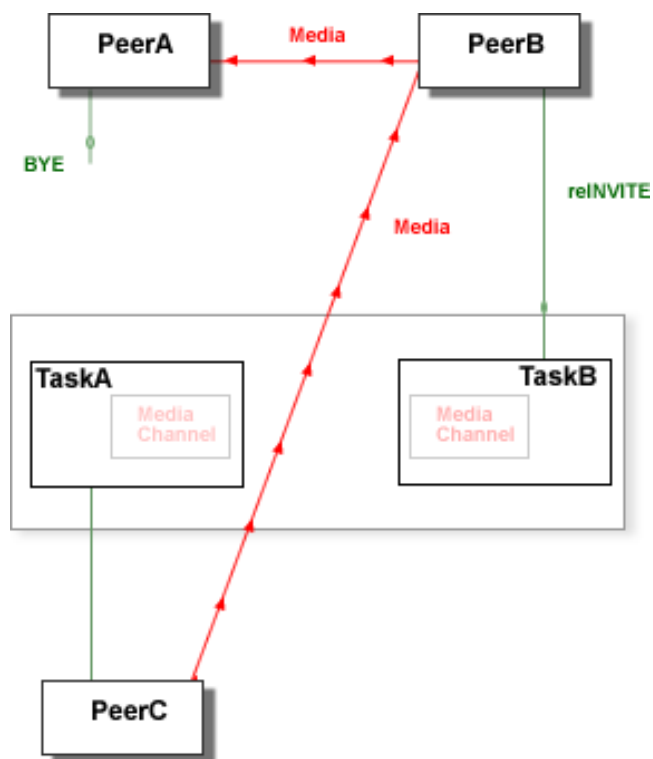
Step 10.



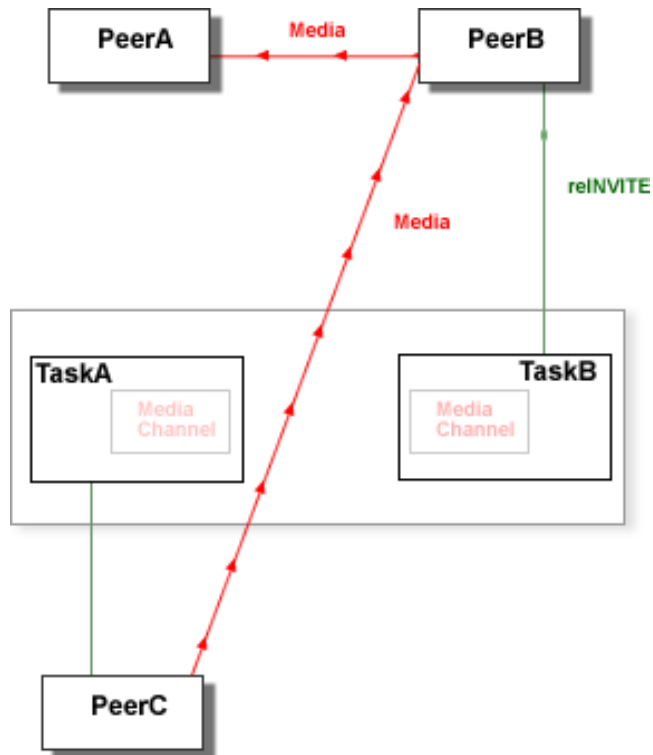
Step 11.



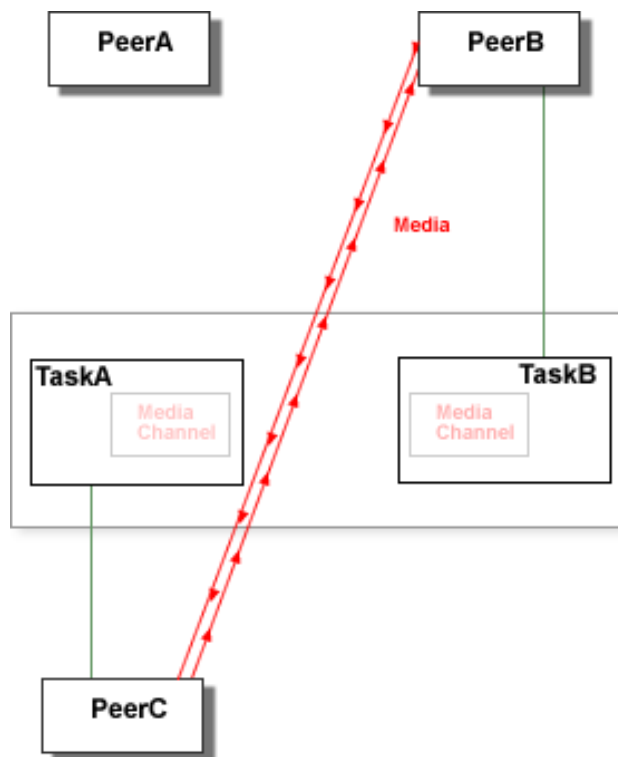
Step 12.



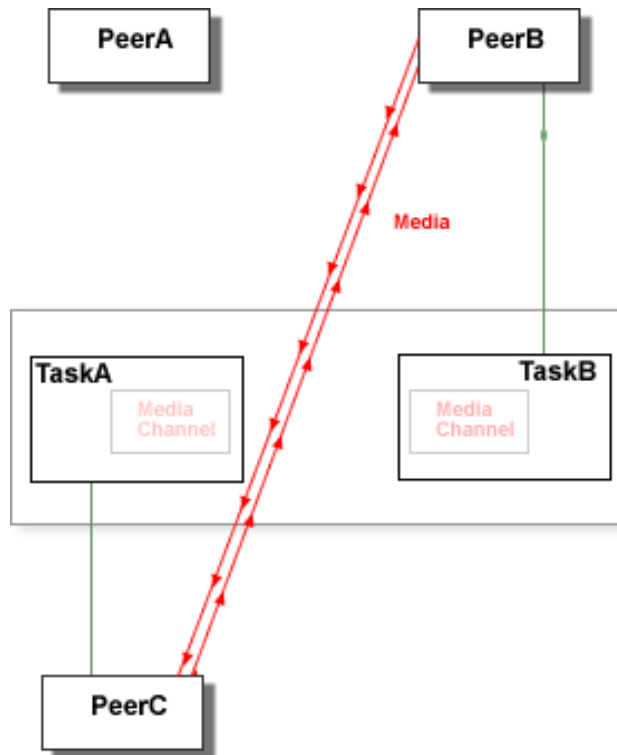
Step 13.



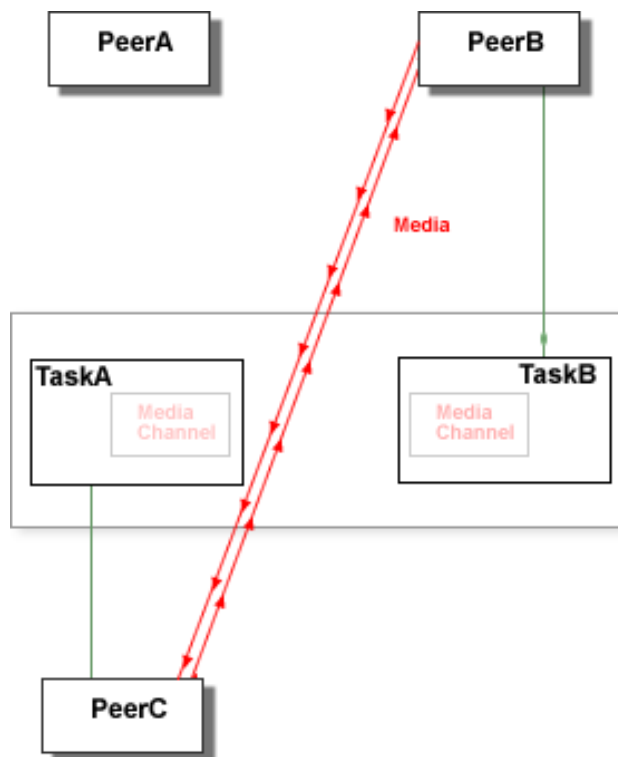
Step 14.



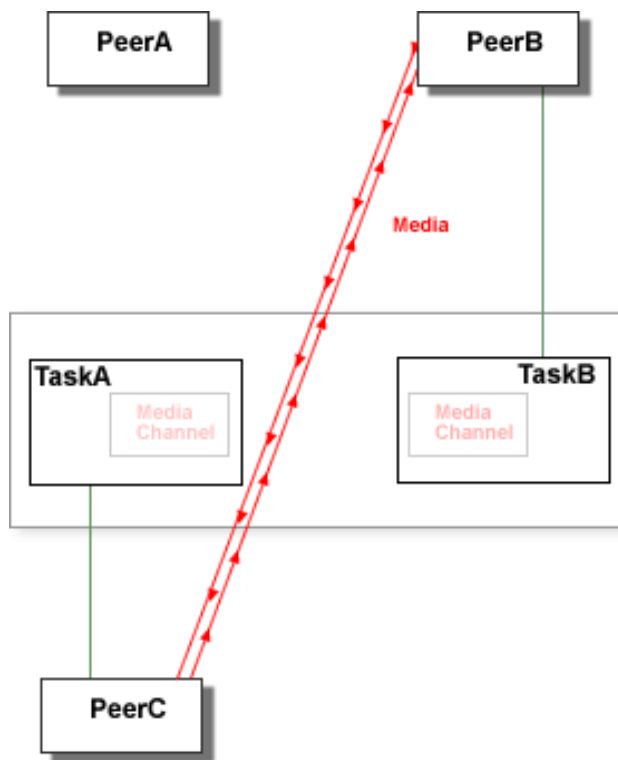
Step 15.



Step 16.



Step 17.



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.

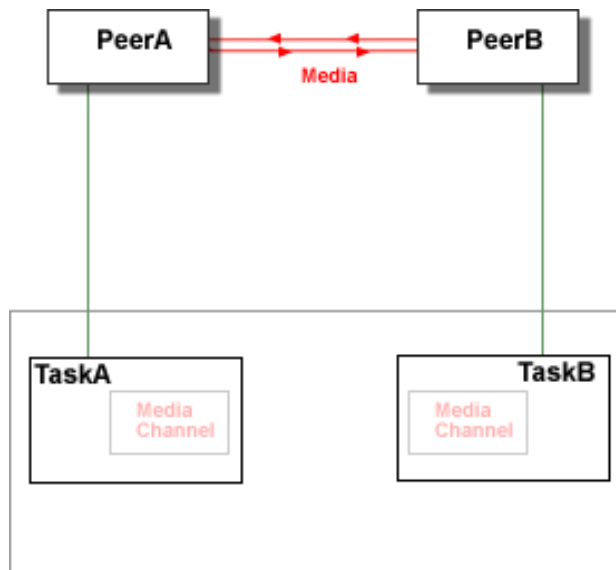
When the Tasks A and B are in the "bridged" state, one of the Tasks (Task A here) can decide to break the Media Bridge:

- an application program the Task A is running uses the [BreakBridge](#) CG/PL function to break the Media

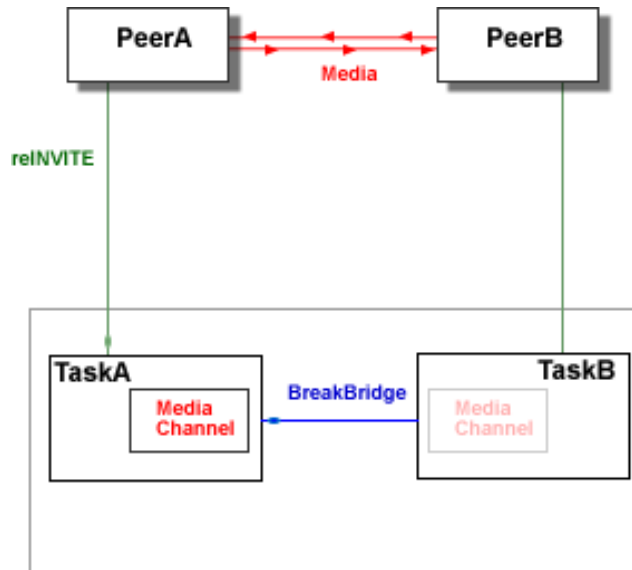
bridge.

- 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.

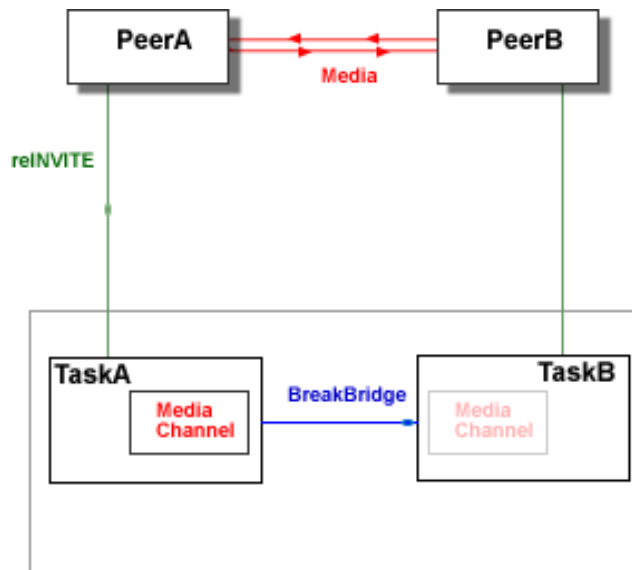
Step 1.



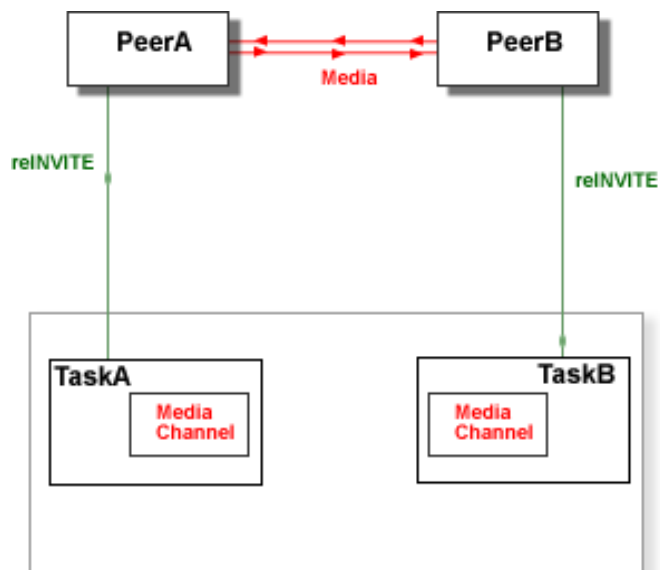
Step 2.



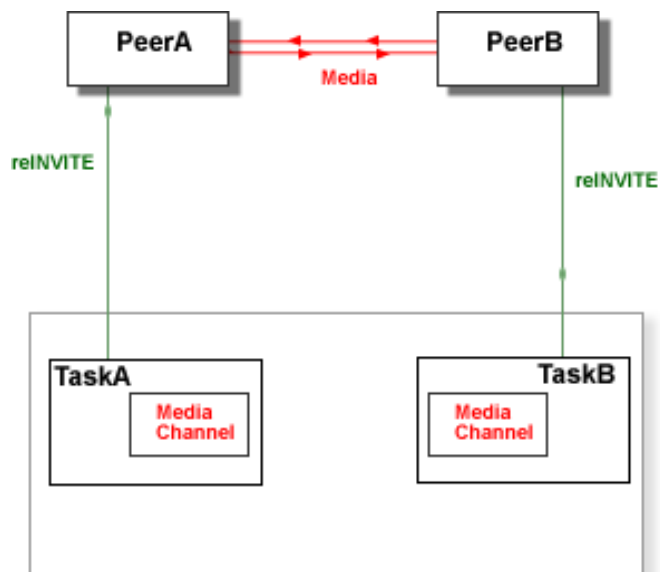
Step 3.



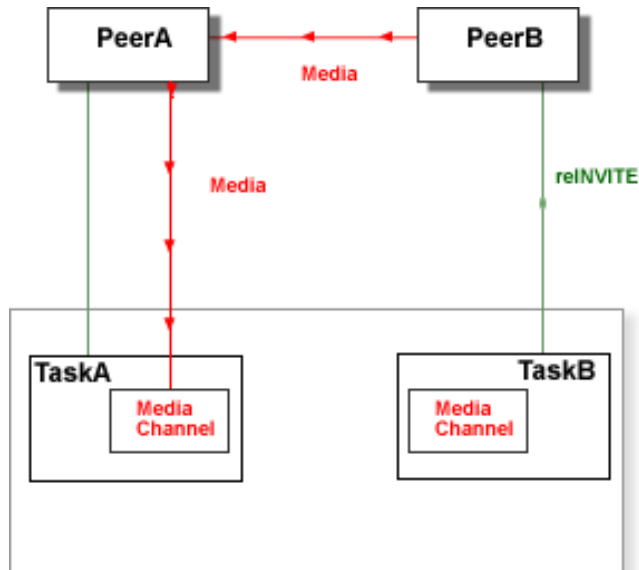
Step 4.



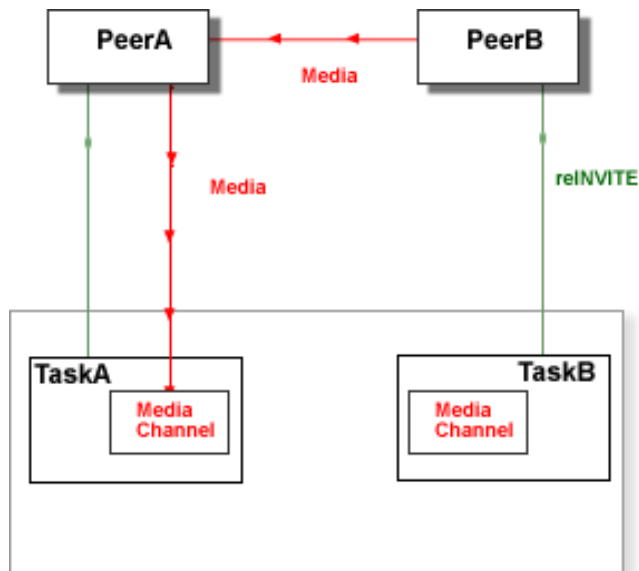
Step 5.



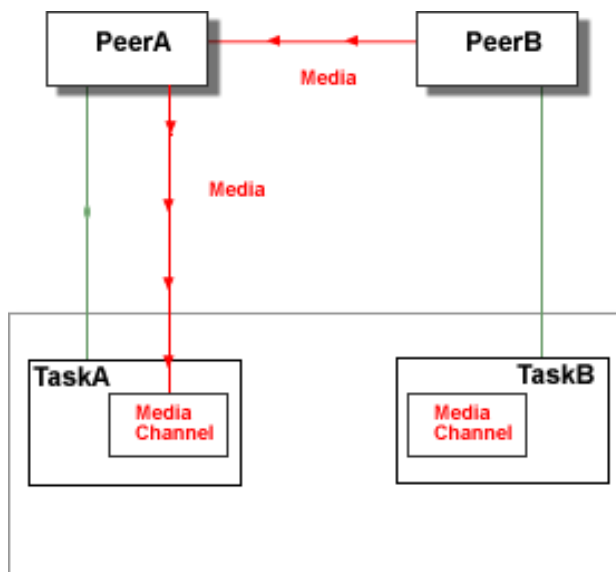
Step 6.



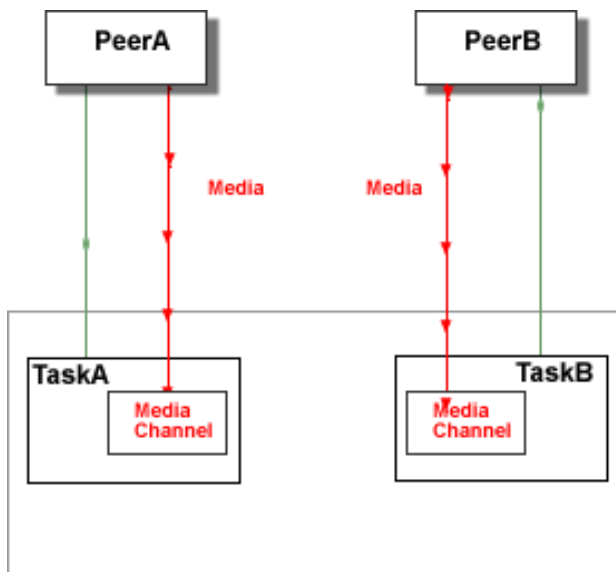
Step 7.



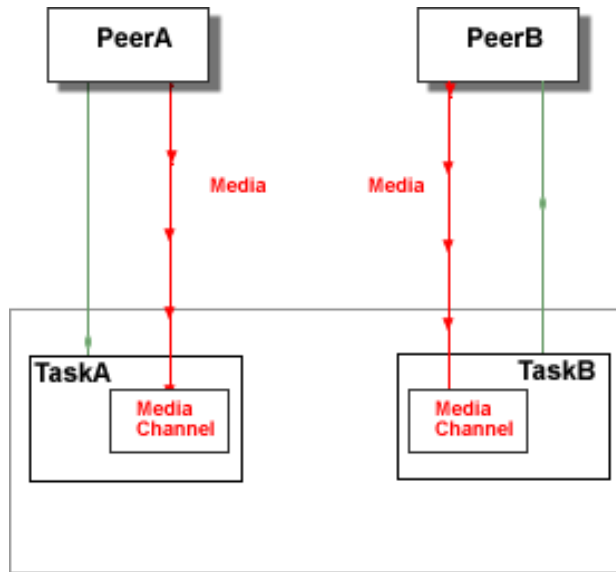
Step 8.



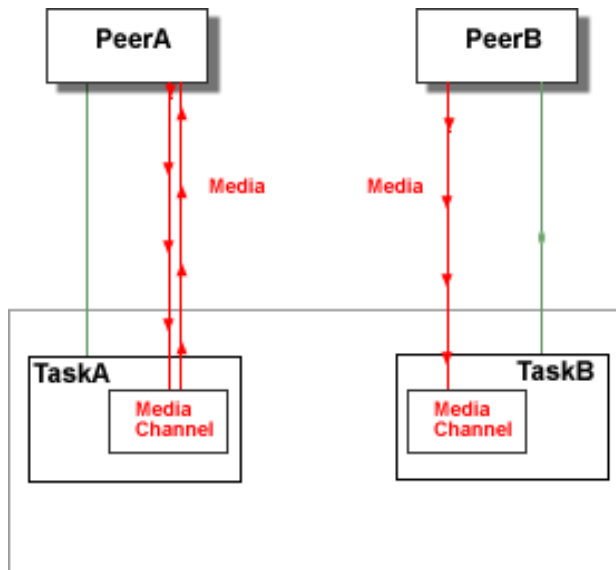
Step 9.



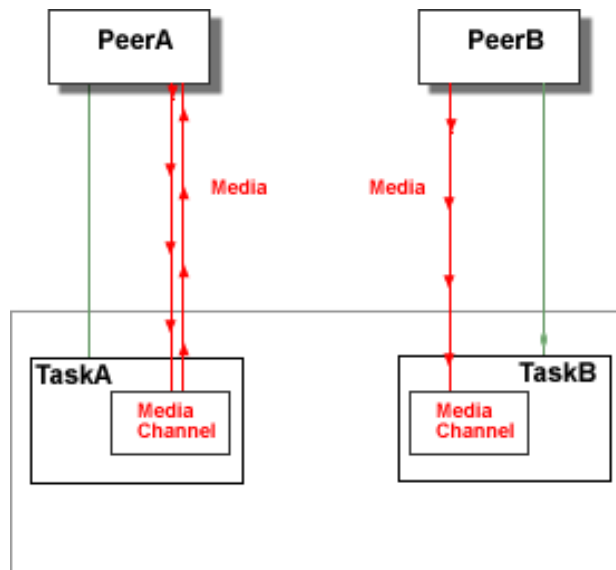
Step 10.



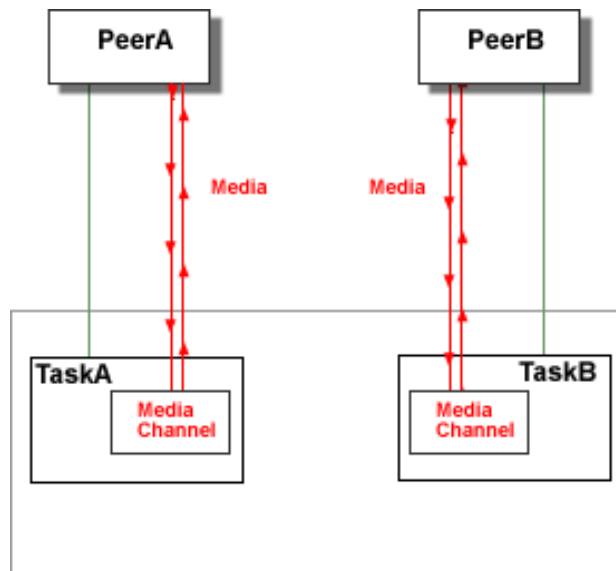
Step 11.



Step 12.



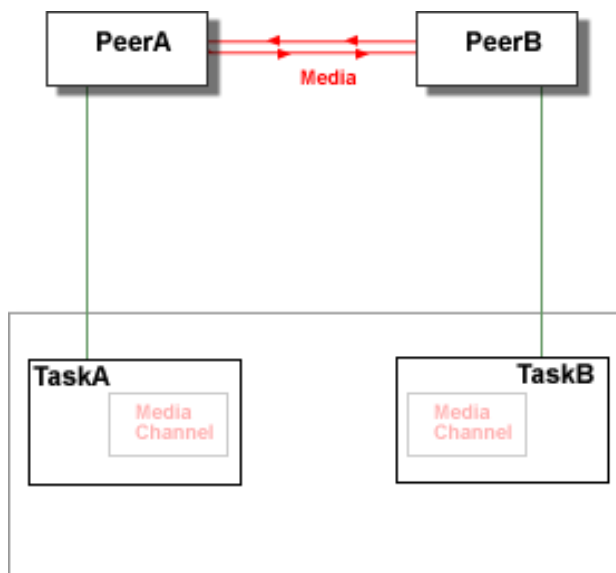
Step 13.



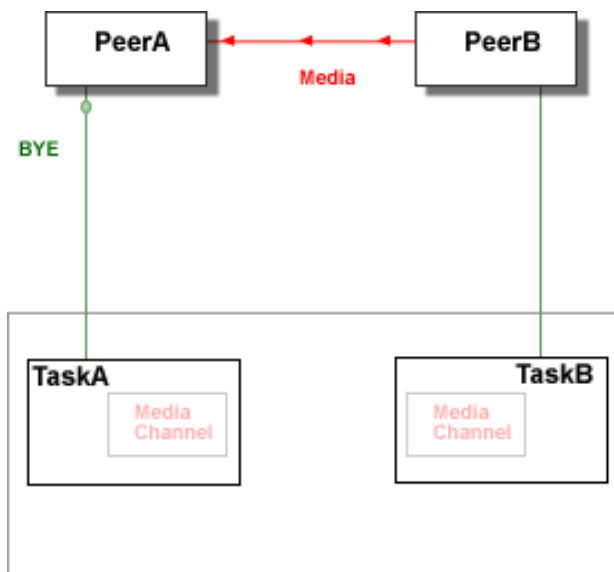
When the Tasks A and B are in the "bridged" state, one of the Tasks (Task A here) can receive a disconnect (BYE) signal from its peer, or it can quit for various reasons. In this case:

- 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.

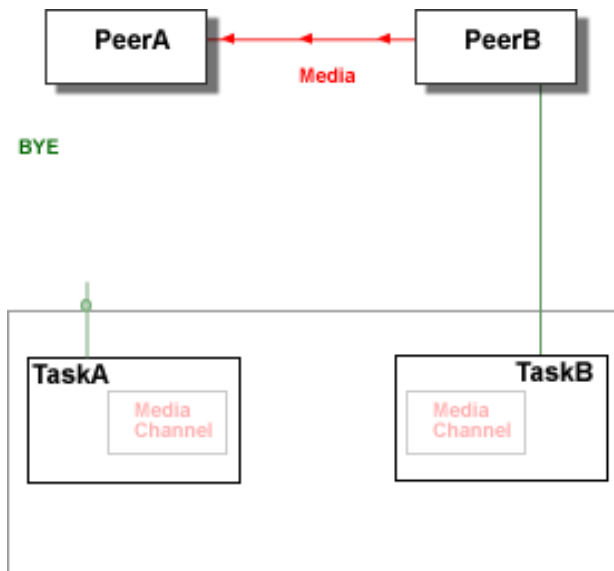
Step 1.



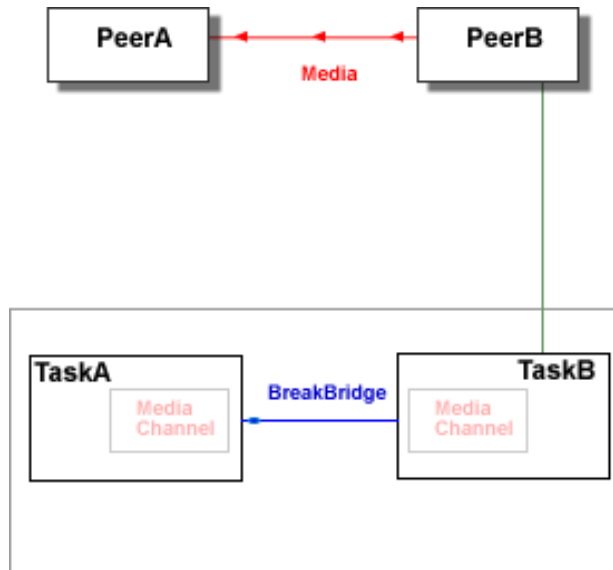
Step 2.



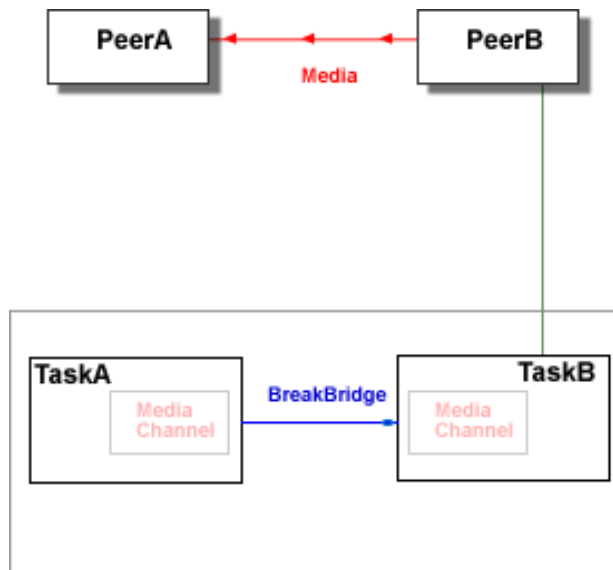
Step 3.



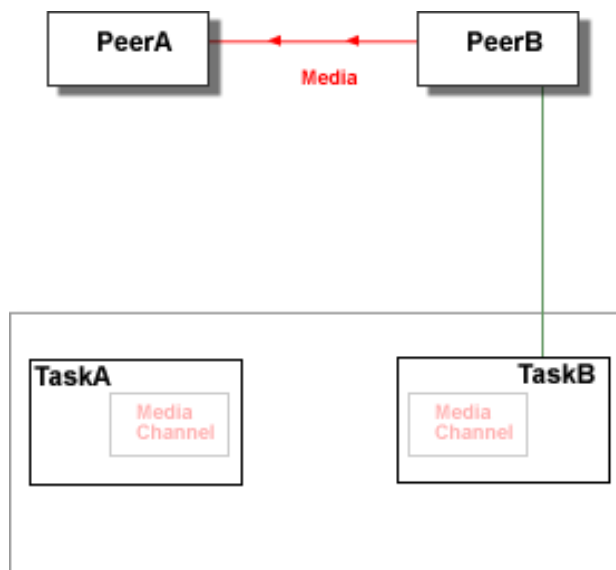
Step 4.



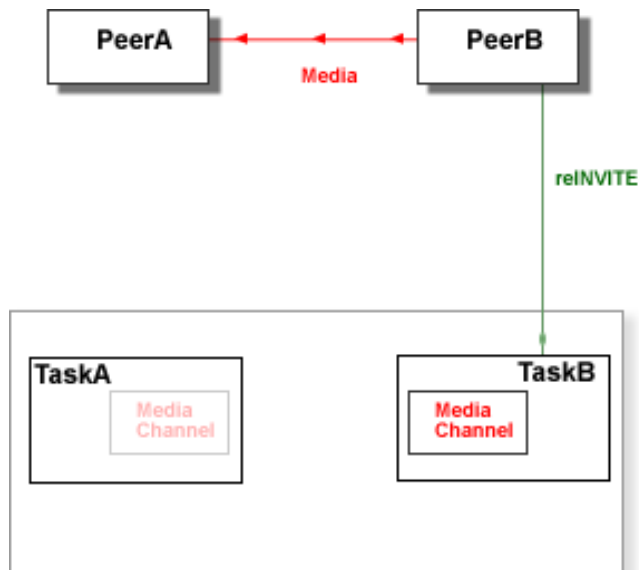
Step 5.



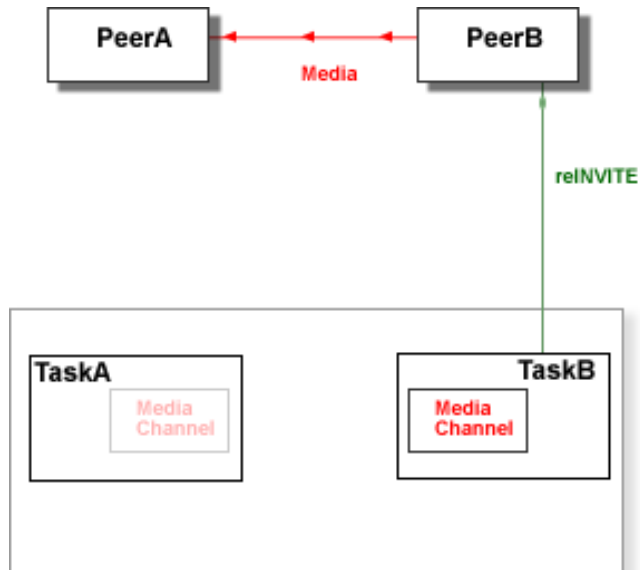
Step 6.



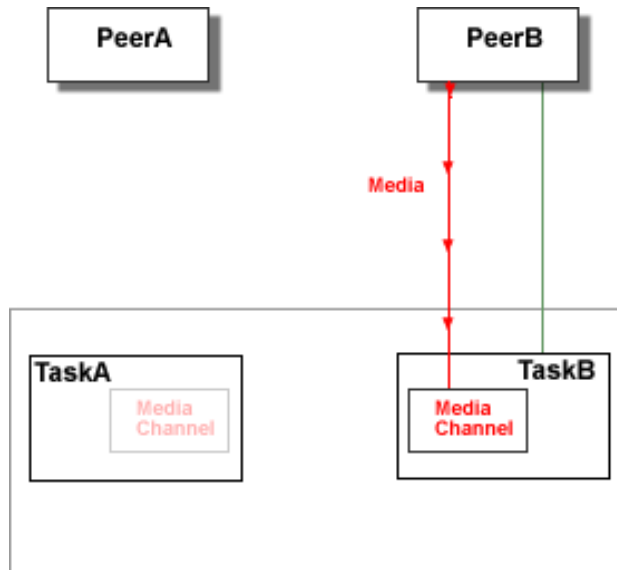
Step 7.



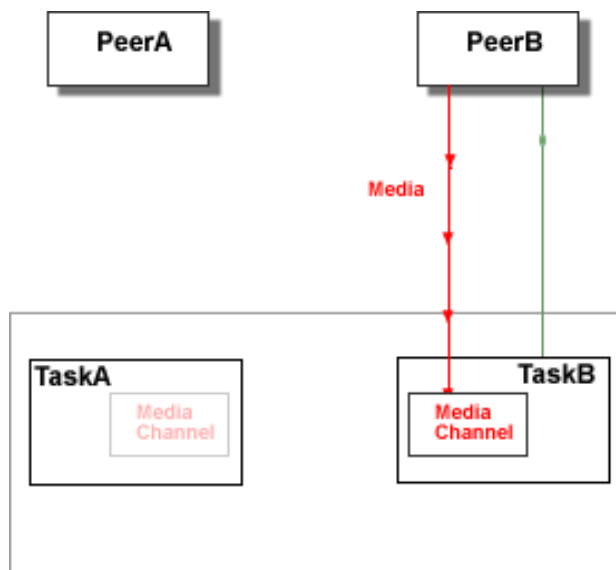
Step 8.



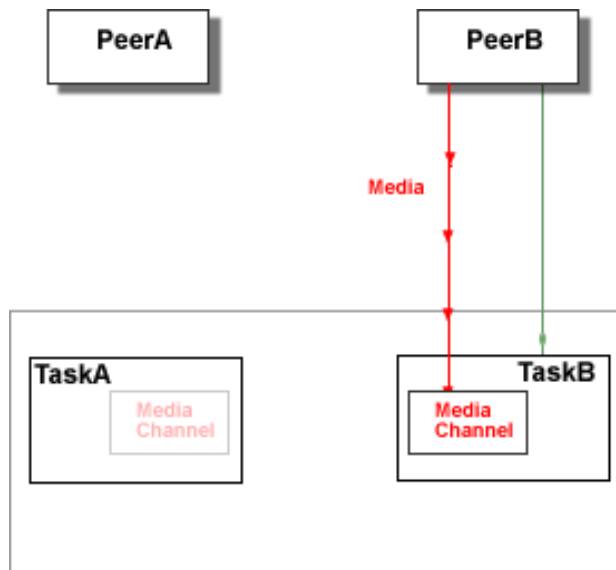
Step 9.



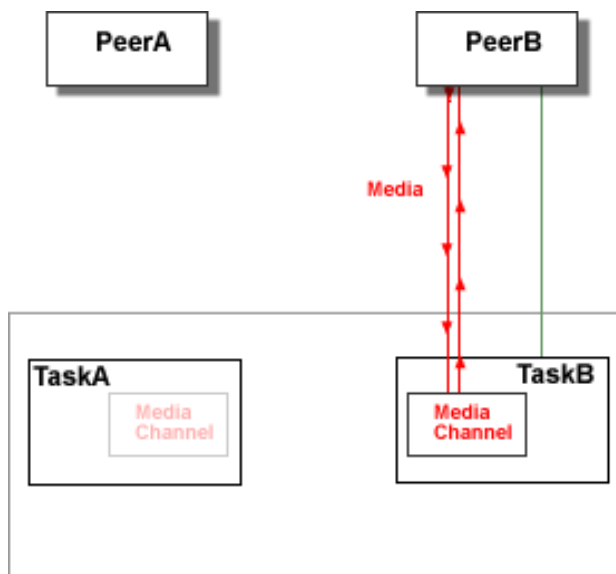
Step 10.



Step 11.



Step 12.



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 signalling 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 `.sppl` extension is loaded from the cur-

rent 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 written in the CG/PL language 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

`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. 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.

The Task is placed into the `disconnected` mode.

`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(startMedia, 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 *startMedia* 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 sus-

pending 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.

```
//
// 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 cancelled, the Task receives a Disconnect Event, and the Task mode changes to *disconnected*.

StartCall(*sendURI*)

This function initiates an outgoing session. The Task should be in the *disconnected* mode.

The *sendURI* 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)
the URI to use for the request From field.
Call-ID (optional)
the string to use for the request Call-ID field.
Remote-Party-Id (optional)
the dictionary to form the request Remote-Party-Id field.

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 been established successfully, 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 signalling 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 an pending outgoing call (initiated using the StartCall function), this procedure cancels that call and the Task will not receive a final response Event.

Disconnect()

This procedure ends the active session, if any, and the Task is placed into the *disconnected* mode.

```
//
// Sample: Connect()/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 not Connect(fromWhom,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.

PendingRequestData(*fieldName*)

This function returns a data element of the pending incoming request.

If no request is pending, the function returns a null-value.

If a request is pending, the functions 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, Remote-Party-Id:

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.

`tag`

the "tag" URI parameter.

`-uri-params`

a dictionary with other URI parameters.

All elements except the address element are optional.

`Route`, `Record-Route`, `Diversion`, `Via`, `Path`, `Supported`, `Require`, `Proxy-Require`,
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 CSeq field.

`User-Agent`, `Reason`

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: for each field value, the array contains 2 strings: the accepted content type and the accepted content subtype. If the field is absent, the function returns a null-value.

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.

`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 a symbol was 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)`

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, [supported extensions](#) are added and tried.

The file should contain media data in one of the [supported formats](#).

Playing 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.

`PlayFileInLoop (fileName, msec)`

This procedure is similar to the `PlayFile` procedure.

The *msec* parameter value should be a number.

This procedure plays the specified file 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 a loop without a predefined time limit.

Playing 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.

`Play (waveData)`

This procedure plays the *waveData* value which should be a datablock.

The datablock should contain media data in one of the [supported formats](#).

Playing 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.

`PlayInLoop (waveData, msec)`

This procedure is the same as the `Play` procedure, but it can play the media data in a loop, in the same way as the `PlayFileInLoop` procedure does.

`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.

```

//
// 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 Personal Site.
//   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 = ReadSiteFile("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 WriteSiteFile("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;

```

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 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)`

This function rejects the *StartBridge* request.

The *input* parameter value should be a *StartBridge* Event to reject.

The *StartBridge* function in the Task that has sent this *StartBridge* Event exits the waiting state and it 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 a *StartBridge* Event to accept.

This function returns a null-value if the Media Bridge was 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.

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 procedure removes the Media Bridge established by 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.

`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 signalling:
// 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
            IsDisconnectedEvent(input) or else input == "#";
    end loop;
    void(BreakBridge());
end procedure;

```

AttachMixer(*input*)

This function takes the peer media of the Task that has sent it a StartBridge request, and attaches it to its own Media Channel.

The *input* value should be a StartBridge Event send to this Task.

This function returns a null-value if the other Task peer media is successfully attached to this Task Media Channel. 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 StartBridge function returns a null-value if that Task peer media is attached successfully. Otherwise, that function returns an error code string.

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 medias 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.

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.

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 signalling.

`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.

Services

`SendVoiceMail(audioData, from, subject, to, cc, bcc)`

This function composes a voice mail E-mail containing the audio data. The *audioData* value should be a datablock containing the data to send.

The *from* value should be a string, it specifies the message `From:` address.

The *subject* value should be a string, it specifies the message `Subject:` header field.

The *to*, *cc*, and *bcc* values should be strings or arrays of strings. These parameters specify the message recipients.

The function returns a null-value if a message was successfully composed and submitted to the Server [Queue](#). Otherwise, the function returns a string containing the error code.

VoiceXML Applications

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.

au (data starts with the .snd tag)

a file should contain PCM 8-bit or 16-bit data, or 8-bit mu-Law data.



XIMSS Protocol

The CommuniGate Pro Server implements the XML Interface to Messaging, Scheduling and Signalling (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.

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.

Examples:

```
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.
- when an asynchronous Server event (such as arrival of an Alert or an Instant Message) is delivered to the client session.

Examples:

```
C:<noop id="A001"/>
```

```
S:<alert>Account is over quota</alert>
```

```
S:<response id="A001"/>
```

```
S:<alert>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.

Login Operation

A client should perform the login operation immediately after a connection to the Server is established.

login

attributes:

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* method only, it contains the user password in the clear-text form.

The Server sends the following data messages:

session

This messages contains information about the newly created session.

attributes:

urlID

the session ID string. This ID string can be used to access session data via [HTTP](#).

Examples:

```
C:<login id="A001" authData="user1@example.dom" password="123rtu" />
S:<session urlID="12-skejlkieuoiuoi-dnciru" />
S:<response id="A001"/>
C:<login id="A001" authData="user1@example2.dom" password="rrr123" />
S:<response id="A001" errorCode="account has been moved to a remote
system" errorNum="518" />
```

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="PDE40TYuNjk3MTcwOTUyQHBvc3RvZmZpY2UucmVzdG9uLm1jaS5uZXQ+" />
C:<auth id="A001"
value="dGltIGI5MTNhNjAyYzdlZGE3YTQ5NWl0ZTZlNzMzNGQzODkw" />
S:<response id="A001"/>
```

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.

`unlockSMIME`

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 `readFolder` 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 Private Key is unlocked for the current session only. If a different session is opened for the same Account, it should unlock the Private Key by itself. The Private Key is automatically re-locked after the specified period of time.

attributes:

`password`

the unlocking string (password).

`duration`

the time (in second) to keep the Private Key unlocked. If this attribute is not specified, the

Private Key is unlocked for 180 seconds (3 minutes).

`modifySMIMEPassword`

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).

`lockSMIME`

This operation locks the Account Private Key stored on the Server.

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:

body:

the Alert text (in the UTF-8 encoding).

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).

`createMailbox`

This operation creates a new [Mailbox](#).

attributes:

`mailbox`

this attribute specifies the new mailbox name.

class

this optional attribute specifies the mailbox class.

renameMailbox

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.

removeMailbox

This operation removes a [Mailbox](#).

attributes:

mailbox

this attribute specifies the existing mailbox name.

children

if this optional attribute is present, all mailbox "children" (nested mailboxes) are removed, too.

listMailboxes

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.

The Server sends the following data messages:

mailbox

These messages are sent when the Server processes the `listMailboxes` request.

attributes:

mailboxName

the mailbox name.

UIDVALIDITY, MESSAGES, UIDNEXT, UNSEEN, INTERNALSIZE, OLDEST, CLASS, MEDIA, UNSEENMEDIA

standard and extended [IMAP](#) mailbox attributes

Mailbox Operations

A client can use the following set of 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`).

`openFolder`

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 is notified about the updates independently, so the client does not need to know that 2 folders are presenting different views on the same Server mailbox.

attributes:

`mailbox`

the mailbox name.

`folder`

the name for the new Folder to be opened. A client can use an arbitrary string as a Folder name. If this attribute is not specified, the mailbox name is used.

`sortField`

the name of a message RFC822 header field to use for mailbox sorting.

the special names `INTERNALDATE`, `UID`, `SIZE`, and `FLAGS` can be used to specify (fast) sorting using the message metadata attributes.

See the [Mailboxes](#) section for more information on the mailbox message flags.

`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 `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.

`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 special names `INTERNALDATE`, `UID`, `SIZE`, `FLAGS` can be used to instruct the Server to retrieve message metadata attributes

Each mailbox can be opened only once. To use a mailbox with a different sorting field or a different *viewer fields* set, close the mailbox, then open it again.

Each session can use several open mailboxes open at the same time.

The client should maintain an internal "view" of the Folder and it should keep that view synchronized with the Server view implemented as a Folder.

Initially the internal view is a table with empty rows.

When a Folder is opened, the Server sends a `folderReport` data message containing the total number of messages in the Folder. This number is used by the client to create the initial internal view Table.

To display the mailbox content on the screen, the client checks which rows of its internal view table it should use. If some of those rows 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).

`browseFolder`

This operation makes the Server send data messages for the specified index (position) interval in an open Folder.

attributes:

`folder`

the Folder name.

`indexFrom`

the index ("view" position) of the first message to send. Specify 0 to retrieve the first message in the view.

`indexTill`

the index ("view" position) of the last message to send.

The Server sends a `folderReport` data message for each position in the specified range. 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 synchronisation model is used. When a message in a Folder is modified, added, or deleted, the client gets a `folderReport` data message with the `mode` attribute value set to `notify`.

The newly added messages do not become visible in the Folder and the deleted messages are not removed from the Folder. Requests to retrieve data from a deleted message return no data items or empty data items.

When the client application is ready to update its internal mailbox view, it should send a `synchFolder` request:

`synchFolder`

This operation tells the Server to send all mailbox modifications to the client.

attributes:

`folder`

the Folder name.

The Server sends `folderReport` data messages. The data message `mode` attribute has the

removed, added, or updated value.

After a `folderReport` notify-mode data message has been sent and before the Server receives the `synchFolder` request, the Server may not send new `folderReport` notify-mode data messages when more changes take place in the mailbox.

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 mailbox.

`copyMessages`

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.

body:

a UID-set (see below).

`markMessages`

This operation modifies [mailbox message flags](#) in an opened mailbox.

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 UID-set (see below).

A UID-set is a set of `UID` XML elements. Each element body is a number - a UID of a mailbox message to use in the request.

`expungeFolder`

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.

`closeFolder`

This operation closes an open Folder.

attributes:

`folder`

the Folder name.

The Server sends the following data messages:

`folderReport`

These messages are sent when a Folder is opened, when a Folder status changes (messages are added, removed, or updated), when the client sends a `browseFolder` or a `synchFolder` request.

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.
This data message is sent when a Folder is being opened.
- 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 `synchFolder` request.
The client should immediately update its internal mailbox 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 `synchFolder` request.
The client should immediately update its internal mailbox 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 the attribute is missing, then the `index` and `UID` attributes specify a mailbox message and the body contains the message *viewer field* values.
- If the attribute value is `notify` some mailbox message(s) were added, modified, or deleted.

The client is expected to send the `synchFolder` request for this Folder.

`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 `init`.

`messages`

the total number of messages in the Folder.

This attribute is present when the `mode` attribute value is `init`, `removed`, or `added`.

`body:`

The body is absent when the `mode` attribute value is `init` or `remove`.

In all other cases, the body contains a set of 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:<openFolder id="A001" folder="INBOX" mailbox="INBOX" sort-  
Field="INTERNALDATE" sortOrder="asc">
```

```
  <field>FLAGS</field><field>From</field><field>Subject</field>  
</openFolder>
```

```
S:<folderReport folder="INBOX" mode="init" messages="234" />
```

```
S:<response id="A001"/>
```

```
C:<browseFolder id="A002" folder="INBOX" indexFrom="0" indexTill="3"  
/>
```

```
S:<folderReport folder="INBOX" index="0" UID="123">
```

```
  <FLAGS>Seen,Deleted</FLAGS><From>John H. Smith</From><Sub-  
ject>Hello - just a test</Subject>  
</folderReport>
```

```
S:<folderReport folder="INBOX" index="1" UID="543">
```

```
  <FLAGS>Seen,Answered</FLAGS><From>Jim Spammer</From><Subject>This  
is the best offer!</Subject>  
</folderReport>
```

```
S:<folderReport folder="INBOX" index="2" UID="343">
```

```
  <FLAGS>Seen,Media</FLAGS><From>&lt;user@example.com>&lt;/  
From><Subject>Meeting reminder</Subject>  
</folderReport>
```

```
S:<folderReport folder="INBOX" mode="notify">
S:<folderReport folder="INBOX" index="3" UID="512">
  <FLAGS>Seen,Flagged</FLAGS><From>&lt;Admin@hq.example.com&gt;</
From><Subject>Shutdown @ 4:45AM</Subject>
  </folderReport>
S:<response id="A002" />
S:<folderReport folder="INBOX" mode="notify">
C:<browseFolder id="A003" folder="INBOX" indexFrom="0" indexTill="3"
/>
S:<folderReport folder="INBOX" index="0" UID="123">
  <FLAGS>Seen,Deleted</FLAGS><From>John H. Smith</From><Sub-
ject>Hello - just a test</Subject>
  </folderReport>
S:<folderReport folder="INBOX" index="1" UID="543">
  <FLAGS>Seen,Answered</FLAGS><From>Jim Spammer</From><Subject>This
is the best offer!</Subject>
  </folderReport>
S:<folderReport folder="INBOX" index="2" UID="343">
  <FLAGS>Seen,Media</FLAGS><From></From><Subject></Subject>
  </folderReport>
S:<folderReport folder="INBOX" index="3" UID="512">
  <FLAGS>Seen,Flagged</FLAGS><From>&lt;Admin@hq.example.com&gt;</
From><Subject>Shutdown @ 4:45AM</Subject>
  </folderReport>
S:<response id="A003" />
C:<synchMailbox id="A004" folder="INBOX" />
S:<folderReport folder="INBOX" index="35" UID="117" mode="deleted"
messages="233" />
S:<folderReport folder="INBOX" index="2" UID="343" mode="deleted"
messages="232" />
S:<folderReport folder="INBOX" index="1" UID="976" mode="added" mes-
```

```
sages="233" >
  <FLAGS>Recent</FLAGS><From>CGatePro Discussions</From><Sub-
ject>[CGP] Re: Session Timer?</Subject>
  </folderReport>
S:<response id="A004" />

C:<browseFolder id="A005" folder="INBOX" indexFrom="0" indexTill="3"
/>
S:<folderReport folder="INBOX" index="0" UID="123">
  <FLAGS>Seen,Deleted</FLAGS><From>John H. Smith</From><Sub-
ject>Hello - just a test</Subject>
  </folderReport>
S:<folderReport folder="INBOX" index="1" UID="976">
  <FLAGS>Recent</FLAGS><From>CGatePro Discussions</From><Sub-
ject>[CGP] Re: Session Timer?</Subject>
  </folderReport>
S:<folderReport folder="INBOX" index="2" UID="543">
  <FLAGS>Seen,Answered</FLAGS><From>Jim Spammer</From><Subject>This
is the best offer!</Subject>
  </folderReport>
S:<folderReport folder="INBOX" index="3" UID="512">
  <FLAGS>Seen,Flagged</FLAGS><From>&lt;Admin@hq.example.com&gt;</
From><Subject>Shutdown @ 4:45AM</Subject>
  </folderReport>
S:<response id="A005" />
```

Example 2:

- **A010:** the Client asks the Server to open 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:<copyMessages id="A010" folder="INBOX" targetMailbox="Archive">
  <UID>512</UID><UID>123</UID><UID>976</UID>
```

```

    </copyMessages>
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 messaging 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:<markMessages id="A020" folder="INBOX" flags="Deleted">
    <UID>512</UID><UID>123</UID><UID>976</UID>
</markMessages>
S:<folderReport folder="INBOX" mode="notify">
S:<response id="A020"/>

C:<synchMailbox id="A021" folder="INBOX" />
S:<folderReport folder="INBOX" index="1" UID="976" mode="updated" >
    <FLAGS>Recent,Deleted</FLAGS><From>CGatePro Discussions</
From><Subject>[CGP] Re: Session Timer?</Subject>
</folderReport>
S:<folderReport folder="INBOX" index="3" UID="512" mode="updated" >
    <FLAGS>Seen,Deleted,Flagged</FLAGS><From>&lt;Admin@hq.exam-
ple.com&gt;</From><Subject>Shutdown @ 4:45AM</Subject>
</folderReport>

```



```
S:<response id="A021"/>

C:<expungeMailbox id="A022" folder="INBOX" />
S:<folderReport folder="INBOX" mode="notify">
S:<response id="A022"/>

C:<synchMailbox id="A023" folder="INBOX" />
S:<folderReport folder="INBOX" index="0" UID="123" mode="deleted"
messages="232" />
S:<folderReport folder="INBOX" index="0" UID="976" mode="deleted"
messages="231" />
S:<folderReport folder="INBOX" index="29" UID="446" mode="deleted"
messages="230" />
S:<folderReport folder="INBOX" index="123" UID="756" mode="deleted"
messages="229" />
S:<folderReport folder="INBOX" index="1" UID="512" mode="deleted"
messages="228" />
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.

`readFolder`

This operation tells the Server to retrieve a Message or its part.

attributes:

`folder`

the Folder name.

`UID`

the message UID.

`totalSizeLimit`

the maximum total size of all message parts returned.

body:

The Server sends the following data messages:

folderMessage

These messages are sent when the Server processes the `readFolder` request.

attributes:

`folder`

the Folder name.

`UID`

the message UID.

body:

The [XML presentation](#) of the Message.

Signalling

A client should use the "bind" operation to start receiving signals directed to the authenticated user.

signalBind

This operation allows the current XIMSS session to receive signals directed to the authenticated user.

attributes:

`deviceName`

an optional string parameter specifying the device used.

signalUnbind

After this operation is complete, the current XIMSS session does not receive signals directed to the authenticated user.

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.

body:

the Instant Message text (in the UTF-8 encoding).

`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`

The Server has received an Instant Message for the client user.

attributes:

peer

the sender E-mail address.

body:

the Instant Message text (in the UTF-8 encoding).

There is no explicit session opening operation. If the client has no open conversation window for the specified user, it should open a new one.

`errorIM`

The Server has failed to deliver an Instant Message.

attributes:

peer

the recipient E-mail address.

`errorText`

the error message text.

`errorNum`

the error message numeric code.

`body:`

the failed Instant Message text (in the UTF-8 encoding).

Preferences

A client can retrieve and update the authenticated user Preferences.

`readPrefs`

This operation retrieves authenticated user Preferences.

`attributes:`

`type`

this 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 attribute is not specified, the effective Account Preferences are retrieved.

`storePrefs`

This operation retrieves authenticated user Preferences.

`attributes:`

`body:`

XML subelements, with the names equal to the Preference element names. The subelement body contains the new Preference element value. If the value is the `default` string, the custom Preference value is removed, and the default value becomes the effective Preference element value. If the value is an array, it should be presented using the `array` XML element.

The Server sends the following data messages:

`prefs`

This message is sent when the Server processes the `readPrefs` request.

attributes:

type

an optional attribute, the same as the request attribute.

body:

XML subelements, with the names equal to Preference element names. The subelement body contains the Preference element value. If the value is an array, it is presented using the [array](#) XML element.

XML Data Formats

Data elements are presented in the XML format using the following conventions.

Arrays

Arrays are presented as a sequence of one or more `subValue` XML sub-elements. The XML subelement body represent an Array element. If an Array is empty, it is presented as one `subValue` XML sub-element without a body.

EMail

This XML element represents an E-mail message or its `message/rfc822` MIME subpart.

attributes:

none

body:

a set of XML sub-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 sub-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`
These elements can contain the `realName` attribute - a MIME-decoded address *name*-

part or *comment*.

The element body is the E-mail address, in the *userName@domainName* format, or the more generic *userName[%domainA[%domainB]]@domainName* format.

If a field contains several addresses, then several XML elements are created, so each element contains exactly one body element.

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 element body is the field global (GMT) time value in the iCalendar format.

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">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 subpart information.

`subtype`

the part Content-Type subtype, if present.

`charset`

the part character set (assume UTF-8 if the attribute is absent).

`ID`

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:

body element(s)

The body element is an optional element. When present, it contains the part body data. The element format depends on the part Content-Type (below the "*" sign 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 the MIME body element).

text/directory, text/x-vcard

Zero or more vCard XML elements.

If vCard parsing failed on one of the part objects, the MIME element contains the `parseError` attribute with a parser error code string.

text/*

Zero or more vCard XML elements.

Note: When a message is being retrieved with the `readFolder` operation, a MIME element body is included only if 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 & 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 & hear a lot!
  </MIME>
  <MIME type="text" subtype="html" charset="iso-8859-1" type-paramX="val-
ueX">
    &lt;HTML&gt;&lt;body BGCOLOR=&quot;yellow&quot;&gt;
    &lt;I&gt;From where I stay&lt;/I&gt;, I can see & hear a lot!
```

```

    &lt;/body&gt;&lt;/HTML&gt;
  </MIME>
</MIME>
</EMail>
vCard

```

This XML element represents a vCard object.

attributes:

modified

This optional attribute contains the value of the REV property (iCalendar-formatted GMT time).

body:

Contains vCard properties stored XML elements with the same names, converted to the uppercase ASCII.

Each property element contains:

attributes:

property parameters stored as element attributes with the same 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.

body:

the property value.

structured values (N,ORG)

the property element body contains a set of subValue XML elements; the each element body is a text element - a subpart of the structured property value.

binary value

the property element body is a base64 XML element; its body is the base64-encoded property value.

regular value (N,ORG)

the property element body contains a text element - the property value.

Example:

```
begin:VCARD
source:ldap://cn=bjorn Jensen, o=university of Michigan, c=US
name:Bjorn Jensen
n:Jensen;bjorn
email;type=internet:bjorn@umich.edu
tel;type=work,voice,msg:+1 313 747-4454
key;type=x509;encoding=B:dGhpcyBjb3VsZCBiZSAkbXkgY2VydGlmaWNhdGUK
end:VCARD
```

<vCard>

```
<SOURCE>ldap://cn=bjorn Jensen, o=university of Michigan, c=US</SOURCE>
<NAME>Bjorn Jensen</NAME>
<N><subValue>Jensen</subValue><subValue>bjorn</subValue></N>
<EMAIL TYPE="internet">bjorn@umich.edu</EMAIL>
<TEL TYPE="work,voice,msg">+1 313 747-4454</TEL>
<KEY TYPE="x509"><base64>dGhpcyBjb3VsZCBiZSAkbXkgY2VydGlmaWNhdGUK</
base64></KEY>
</vCard>
calEvent
```

This XML element represents a VEVENT calendaring object.

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 session 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

where

sessionID

the current XIMSS session ID, sent with the session data message.

folder

the target Folder name

UID

the target message UID in the target Folder

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 readFolder command. 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-format-
ted messages.

Example:

```
C:GET /Session/1-2xklkdld8-djdkjk/MIME/INBOX/567-01-B.gif HTTP/  
1.1
```

Attachment Uploading



Web Application Module

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 Web User 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 Web User 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 specify 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 if 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.

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 used 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 [Code Components](#) section 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

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 Domains realm of the WebAdmin Interface, and click the Skins link.

To manage the Domain Skins, open that Domain page in the Domains 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:

Marker	File Name	Size	Modified
<input type="checkbox"/>	Help.gif	155	25-Sep-04
default	addressbook.wssi	1143	23-Sep-03
default	alerts.wssp	1727	26-Sep-04
default	answeredletter.gif	890	27-Feb-03
default	attachedFile.gif	1147	27-Feb-03
	...		
default	mailbox.wssp	5806	28-Sep-04
<input type="checkbox"/>	mailboxes.wssp	3452	02-Oct-01
	...		
default	strings.data	28K	27-Oct-04
default	german.data	31K	28-Oct-04
	...		
default	website.wssp	592	28-Sep-04
default	websitebody.wssi	2648	28-Sep-04
Delete Marked		Upload File:	

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 file first. If the compiler parser detects an error, the file is not uploaded, the source of the file 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	
Marker	Skin Name
<input type="checkbox"/>	GoldenFleece
<input type="checkbox"/>	IceColdMail
<div><div>Delete Marked</div><div>Create Skin</div><div></div></div>	

To create a Named Skin, enter its name, and click the Create Skin 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.



WSSP Scripting

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 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 signs (%%)

- 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
- logical expression

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 `NULL`.

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 array scanner name. It can be used only inside the `<!--%FORALL name...-->...<!--%ENDFOR name-->` structure where this array element is defined (see below). The array 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 sign specifies the key to be used to extract the value from that dictionary. If the value of the expression before the dot sign is not a dictionary, or if it does not contain the specified key, the keyed element value is `NULL`.

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 `NULL`.

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. The *index expression* is calculated, and its value should be a string representing a number. This number specifies which array element becomes the value of this indexed expression. The first array element is retrieved if the value of the *index expression* is 0.

If the value of the expression before the bracket symbol is not an array, or if the value of the *index expression* is not a string, or if the value of the *index expression* represents a number that is negative or is equal or greater than the number of array elements, the value of the indexed element expression is `NULL`.

An alphanumeric string followed by the `(` symbol is a function call. Elements after the `(` symbol specify the function parameters, and they are followed by the `)` symbol.

Function names are case-insensitive.

The following list specifies the available functions and their parameters.

`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 quoted string.

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: <i>accountName@domain-Name</i>
loginAddress	a string specifying the network (IP) address the user was using when initiating this session
loginTime	a string with the session start time in the ACAP format
mailboxes	an array with the names of "selectable" mailboxes
selectedMailbox	a string with the name of the target mailbox for the last Copy/Move operation
webFolders	an array the Personal File Site folder names
selectedWeb-Folder	a string with the name of the target File Site folder for the last Store File In operation
webSiteEnabled	this YES string element exists if the storage limit for the Personal File Site is not set to zero

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 quoted string.

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 NULL.

EXISTS (*expression*)

The parameter is an expression. Its value is calculated, and the function returns the string YES if the calculated value is not NULL, or the string NO if the returned value is NULL.

DOESNOTEXIST (*expression*)

The parameter is an expression. Its value is calculated, and the function returns the string NO if the calculated value is not NULL, or the string YES if the returned value is NULL.

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.

NOT (*expression*)

The parameter is an expression. Its value is calculated, and the function returns NULL if the calculated value is a string that does not start with one of the symbols N, n, -, or 0, 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 NULL), the function returns the string YES. Otherwise the function returns NULL.

EQUALSNOCASE (*expression1* AND *expression2*)

Both expressions are calculated. The function returns the string YES if both calculated values are NULL or if both values are strings and these strings match using the ASCII case-insensitive comparison opera-

tion. In all other cases the function returns NULL.

`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 NULL.

`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 NULL.

`ISINDEX(expression IN scanner)`

The *scanner* should be the name of the `<!--%%FORALL scanner IN ...-->` construct surrounding the current portion of the script code. The expression value is calculated, and if it is a string and 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 NULL.

`ISFIRST(scanner)`

The *scanner* should be the name of the `<!--%%FORALL 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 NULL.

`ISEVEN(scanner)`

The *scanner* should be the name of the `<!--%%FORALL scanner IN ...-->` construct surrounding the current portion of the script code. If the current value of the array index is even, the function returns the string YES. Otherwise the function returns NULL.

`CHECKED(expression)`

The parameter is an expression. Its value is calculated, and the function returns the string CHECKED if the calculated value is a string that does not start with one of the symbols N, n, -, or 0, otherwise the function returns NULL.

`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 NULL.

`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 NULL.

`ISNUMBER(expression)`

The parameter is an expression. Its value is calculated, and the function returns the string YES if the cal-

culated value is a number. Otherwise the function returns NULL.

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 NULL.

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 NULL.

ISDATE (*expression*)

The parameter is an expression. Its value is calculated, and the function returns the string YES if the calculated value is a date-type object. Otherwise the function returns NULL.

ISDATA (*expression*)

The parameter is an expression. Its value is calculated, and the function returns the string YES if the calculated value is a binary-data-type object. Otherwise the function returns NULL.

NULL ()

The value of this function is NULL.

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.

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 NULL. 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 NULL. 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 NULL. 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 `NULL`.

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, etc.)

`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, etc.)

`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 NULL.

Logical expressions are two expressions separated with a | (OR) sign or with a & (AND) sign. You can also use parentheses to enclose expressions:

```
expression1 | expression2
expression1 & expression2
(expression1 & expression2) | expression3
```

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 value is not a string, then the entire text element is removed from the resulting markup code. If the result is a string, it substitutes the text element in the resulting markup code.

`%%HTML: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 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 expression result is the `>=GO=>` string, the text element is substituted with the `>;=GO=>;` string.

`%%HTMLUTF8: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 substitutes the text element using HTML escape symbols (the string is not converted from the UTF-8 charset into the page charset).

Example: if the expression result is the `>=GO=>` string, the text element is substituted with the `>;=GO=>;` string.

`%%URL: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 substitutes the text element using URL escape symbols.

Example: if the expression result is the `Stop It?` string, the text element is substituted with the `Stop%20It%3F` string.

`%%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, then the entire text element is removed from the resulting markup code. If the result is a string, it substitutes the text element using escape symbols needed for JavaScript strings. The result string is converted into Unicode and its non-ASCII symbols are presented using the `\uhhhh` 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 the `What do \"they\" think` string.

`%%SIZE: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 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.

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 `NULL`, 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 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 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 `NULL`, 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 40 minutes string.

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 "date" type, then the entire text element is removed from the resulting markup code. If the result is a "date" element 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 date 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 suffix, 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 added 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 date value is converted into the local time first.

`%%DATETIMEAS (format) : expression%%`

The expression is calculated. If the value is not of the "date" type, then the entire text element is removed from the resulting markup code. If the result is a "date" element 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 date 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 date specified with the expression is "very close" to the current date (the absolute difference is less than 22 hours),
- the `DATETIME ("monthDate")` prefix if the date 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 & 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 `<!--%%FORALL 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 `<!--%%FORALL 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.

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 can be followed with the

`<!--%%ELSE-->`

element, and must be followed with the

`<!--%%ENDIF-->`

element.

The value of the expression is calculated, and if it is a string and that string does not start with one of the symbols N, n, -, or 0, the portion of the script between the `<!--%%IF expression-->` element and the `<!--%%ELSE-->` element is processed. If the `<!--%%ELSE-->` element does not exist, the portion of the script between the `<!--%%IF expression-->` element and the `<!--%%ENDIF-->` element is processed.

If the *expression* result is not a string, or it is a string that starts with one of the listed symbols, the portion of the script between the `<!--%%ELSE-->` element and the `<!--%%ENDIF-->` element is processed. If the `<!--%%ELSE-->` element does not exist, the portion of the script between the `<!--%%IF expression-->` and `<!--%%ENDIF-->` elements is removed completely.

Example:


```
<!--%%IF EXISTS(lastLogin)-->We have not seen you since  
<I>%%HTML:lastLogin%%</I>  
<!--%%ELSE-->Welcome, new user!  
<!--%%ENDIF-->
```

In this example, if the dataset contains the `lastLogin` element with the `20-Apr-2001` string value, this script portion will produce

```
We have not seen you since <I>20-Apr-2001</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.

The portion of the script between the `<!--%%FORALL scanner...-->` and `<!--%%EMPTYFOR scanner-->` elements or (if the `<!--%%EMPTYFOR scanner-->` element does not exist) the portion of the script between the `<!--%%FORALL scanner...-->` and `<!--%%ENDFOR scanner-->` elements is processed repeatedly, for each array element.

Expressions specified in that portion of the script can use the `scanner[]` array *scanner* reference to access the current element of the *expression* array value.

If the *expression* value is not an array or if it is an empty array, and the `<!--%%EMPTYFOR scanner-->` element is specified, the portion of the script between the `<!--%%EMPTYFOR scanner-->` and `<!--%%ENDFOR scanner-->` elements is processed. If the resulting array has at least one element, this portion is not processed.

Example:

```
<TABLE BORDER=1>
```

```

<TR><TD>File Name</TD><TD>File Size</TD></TR>
<!--%%FORALL elem in fileList-->
<TR><TD>%%HTML:elem[].name%%</TD><TD>%%elem[].size%%</TD></TR>
<!--%%EMPTYFOR elem-->
<TR><TD colspan=0>&nbsp;</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 `<!--%%FORALL ...--> ...<!--%%ENDFOR ...-->` constructs can be nested.

```
<!--%%INCLUDE filename[( parameter1 [, parameter2 ... ])]-->
```

The file with the specified *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, not a pre-processor operator. As a result, if an `<!--%%INCLUDE filename-->` element is used within a `<!--%%FORALL ...--> ...<!--%%ENDFOR ...-->` construct, the *filename* code can be executed several times, once for each element of the array used in the `FORALL` construct.

```
<!--%%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 DICTIO-  
NARY("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 `NULL`, 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
(number1,number2,...,numberN) [ DISPLAY dictionary]-->
```

This element is processed in the same way as the `NUMERICMENU` element. The *numberX* numbers specify time intervals in seconds.

The *selected* and *selectedDefault* expressions should return strings. Each string is converted into a numeric value (the number of seconds) using the algorithm used in the text elements with the `TIME` : prefix.

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  
(number1,number2,...,numberN) [ DISPLAY dictionary ]-->
```

This element is processed in the same way as the NUMERICMENU element. The *numberX* numbers specify a data size in bytes.

The *selected* and *selectedDefault* expressions should return strings. Each string is converted into a numeric value (the number of bytes) using the algorithm used in the text elements with the SIZE : prefix.

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 dic-  
tionary ]-->
```

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 NULL, 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 in the HH:MM format, where HH is the hour, and MM is the minute.

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 have a date-type value. 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 have a date-type 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] [DAYS AFTER  
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 date-type object, 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* date value. If the selected value is equal to the *remote future* value, a menu control is composed.

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] [DAY-  
SAFTER after] -->
```

This element creates the same markup code as the CALENDARDATECONTROL element.

The *selected* expression value should have a date-type value. The value is assumed to be expressed in the global (GMT) terms, so the value is converted to the local time first.



Web Application Code Components

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 parame-

ter 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 `` HTML element will display the `Logo.jpeg` 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` `type:string`

This element value is a string with the name of the CommuniGate Pro Domain. 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](#).

`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 the request is a WML request, the `wlogin.wssp` page is used.

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* (*wStartPage* for WML sessions) Skin string.

If login operation was successful, but the request contained the *restoreSessionPage* parameter, the *resume.wssp* (*wresume.wssp* for WML sessions) 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 then 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 Lists](#).

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 `discon-`

nected.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 `SMIME-Password` 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.

Example: The Stock Skin uses `Hello.wssp` 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, wMailboxes

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 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 to the element 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.

`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 account does not have the total mailbox size limit, this element contains the `unlimited` string.

Name: Mailbox, wMailbox

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 Web User 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" 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 `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`, `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, wContacts

Processed in the same way as the Mailbox page.

Name: Notes, wNotes

Processed in the same way as the Mailbox page.

Name: Calendar, wCalendar

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 Web User 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 substracted 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 ele-

ment 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 14-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 14-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, wTasks

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 Web User 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 a Personal File Site folder, the file parts of the message (attachments, images) are stored in the specified File Site 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 based 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 `desired-Charset` 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.

isEvent

This element with the `Yes` value is added to the result dataset if the item being composed is a Calendar Task (ToDo) item.

isEvent

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 *annn* and *mnnn* 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 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 ($1 - \text{current sorting order}$).

`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 ($1 - \text{current sorting order}$).

`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 `YES` string element exists if the column is the Sent or Received 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 if 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 sent to the client browser. Before these documents are sent, their first lines are checked. If a document 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 processed 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.



WebUser Interface (WebMail)

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 the 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 files in your Personal [File Site](#).

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:

**Welcome to CommuniGate Pro,
the domain1.dom Messaging Server!**

Registered Users

Login Name	Password		
<input type="text" value="john.smith"/>	<input type="password" value="****"/>	<input type="button" value="Enter"/>	<input type="checkbox"/> Disable Network Address check
			<input type="checkbox"/> Disable Cookie check

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 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. 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

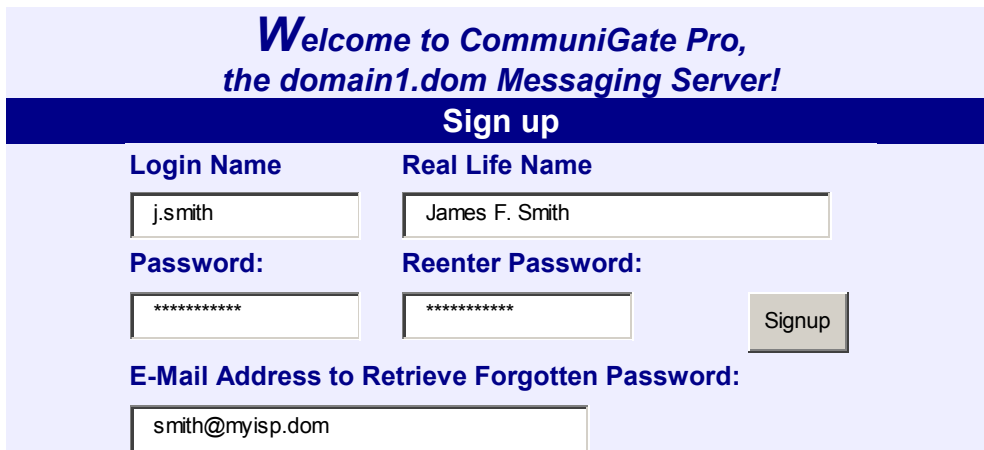
`http://yourserver:port/imode`

To use the Japanese version of the IMode interface, point your device browser to

`http://yourserver:port/imodejp`

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:



The screenshot shows a web form titled "Welcome to CommuniGate Pro, the domain1.dom Messaging Server!". Below the title is a dark blue bar with the text "Sign up" in white. The form contains several input fields and a button:

Login Name	Real Life Name
<input type="text" value="j.smith"/>	<input type="text" value="James F. Smith"/>
Password:	Reenter Password:
<input type="password" value="*****"/>	<input type="password" value="*****"/>
E-Mail Address to Retrieve Forgotten Password:	
<input type="text" value="smith@myisp.dom"/>	

A "Signup" button is located to the right of the password fields.

The passwords entered in both passwords fields should match.

You may want to enter the Forgotten Password Retrieval Email - 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](#). Besides, it contains 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) ▼
Require Fixed Network Address:	Yes ▼
Use Cookies:	Yes ▼
Time Zone:	*** ▼

Layout:

Use this setting to specify the Layout ("Skin") of the WebUser Interface. Select the *** 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 possess 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 will use the select time zone to show the date and time values. Select the *** value to use the Server OS time zone.

Text Encoding	
Preferred Character Set:	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:

Password Modification	
Current Password:	<input type="password" value="*****"/>
New Password:	<input type="password"/>
Reenter New Password:	<input type="password"/>
Forgotten Password Recovery	
E-Mail Password to:	<input type="text" value="john.doe@domain.co"/>
<input type="button" value="Modify"/>	

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.

Name Value	
Work Phone:	+1 415 383 7164
City:	Mill Valley

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.

Automated Rules

The WebUser Interface provides access to the account [Automated Mail Processing Rules](#). If the `Can Modify Account Rules` [account option](#) is not enabled, then you can view the account 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 the `Can Modify Account Rules` option is not enabled for your account.

See the [Automated Mail Processing Rules](#) section to learn how to specify the Rules.

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:	Move To Trash ▼
Trash Mailbox:	default(Trash Can) ▼
Keep Message Received Time:	Yes ▼
On Logout Remove from Trash if Older than:	2 days ▼

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 logout of the WebUser Interface, the system 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.

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 SMIME [Service](#) enabled.

See the [WebUser Interface SMIME](#) section to learn how to use Secure Mail functions.



WebUser Interface: Mailboxes

One of the main functions of the CommuniGate Pro WebUser Interface is Web 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

The Mailboxes page displays your mailboxes. It allows you to open the listed mailboxes and to create a new mailbox:

Create	<input type="text"/>	<input type="text"/>	used 238K of 3M	
Empty Trash			254K in Trash	
Display	Filter: <input type="text"/>		3 selected	
Mailbox	Size	Messages	New	Unread
INBOX	20K	5	2	3
Friends	200K	56	4	4
Friends/Family	18K	3		

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:

Display All Account Mailboxes:	<input type="text" value="Yes"/>
Display Subscribed Mailboxes :	<input type="text" value="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:

Display
100
Filter:
Search:

	Status	From	Subject	Size	Received
<input type="checkbox"/>		TestList administration	Welcome!	871	25-Nov-03
<input type="checkbox"/>		Technical Support	Fwd: [*] CommuniGate Pro 4.2b2 released	4K	25-Nov-03
<input type="checkbox"/>		Technical Support	TEST - text & 2 gifs	11K	25-Nov-03
<input type="checkbox"/>		Technical Support	Fwd: TEST - text & 2 gifs	13K	25-Nov-03
<input type="checkbox"/>		philip@node5.stalker.com	(no subject)	5K	25-Nov-03
<input type="checkbox"/>		Technical Support	FWD:(no subject)	5K	25-Nov-03
<input type="checkbox"/>		Technical Support	Fwd: HTML letter (alternative) (no subj	6K	25-Nov-03
<input type="checkbox"/>		John R. Smith	Re: Weird Problems	2015	03-Dec-03
<input type="checkbox"/>		Douglas M.	bad log file from our server	8K	11-Dec-03
<input type="checkbox"/>		U&B	LDAP	1575	21-Dec-99
<input type="checkbox"/>		James Green	design suggestion	3K	23-Dec-03
<input type="checkbox"/>		Adam Drake	Transmission problems between SIMS and	4K	05-Jan-04

Read: Set Clear
 Flagged: Set Clear
 Deleted: Set Clear
Purge Deleted

Copy to... Move to...
-- select mailbox --
[Mailbox Management](#)

Redirect to... Forward to...

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 can be 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:

Mailbox Viewer		Display: 100	Refresh Every: 5 minutes				
Default	Fields:	Status	From	Subject	Size	Received	Reverse
	Sort:	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	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 `-nothing-` and click the Update button to remove a field.

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:

Miscellaneous	
Message Delete Method:	Move To Trash

Move To Trash

The delete operation moves the selected message(s) to the Trash mailbox. This option is 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 cer-

tain rights from a particular user, "grant" those rights to the identifier `-username`. See the [Mailboxes](#) section for more details.

Access Control List									
Identifier	Lookup	Select	Seen	Flags	Insert	Post	Create	Delete	Admin
<input type="text" value="-badguy"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="text" value="anyone"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="text" value="susan"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

If you select the Apply to Sub-Folders option, then the selected Access Control List 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:

Folder Subscription
Drafts
Sent
~sales/INBOX
~support/Pending

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.

Folder Aliases	
Alias Name	Mailbox Name
salesBox	~sales/INBOX

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.

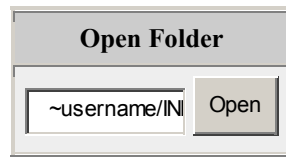
Your mail client software will list the created mailbox aliases as well as the real mailboxes created in your account. You can open a mailbox alias as you open a real mailbox, and the specified foreign mailbox will be opened.

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:





WebUser Interface: Messages

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: Manually ▼	Show Links: default (safely)	Show Images: default (safely) ▼	
Custom ▼ Fields: From Subject Date To X-Mailer			

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 Mailbox page code. If you use a browser that does not support embeded frames (Netscape 4.x), 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 Mailbox page code. The WebUser Interface checks the message HTML portion code and removes some tags that may distort the entire WebUser Mailbox page.

disabled

In this mode the PlainText portions of messages 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 "refer-

rer" 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 references are modified, so Web sites hosting external images do not receive the "referrer" information containing your WebUser Session ID.

`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

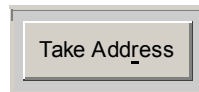
When you open a message with a Read Notification request for the first time, a confirmation (Read Receipt) message is automatically sent to the message author.

Fields

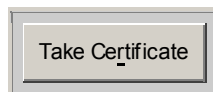
The fields listed in this set are displayed in the message headers.

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:



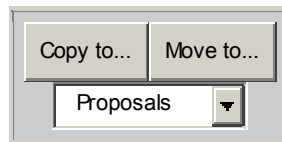
If a message contains a *digital signature* and your CommuniGate Pro Account has the Secure Mail feature enabled, you can add the sender's *certificate* (also known as *digital ID*) to your Address Book:



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).



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.



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 Personal File Site feature is enabled for your CommuniGate Pro Account, you can store attachments and embedded images directly to your File Site folders.

When a message has file parts, the Store In button appears:



Select a File Site 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.



WebUser Interface: Composing Messages

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 of 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.

Message Composer		Auto Wrap: default (Yes)	Text columns: 72
From Address:		MIME-encode Headers: default (Yes)	
Signature:	<div>Sincerely, John</div>		
Store: <input checked="" type="checkbox"/>	Sent Messages in	<div>Sent</div>	<input checked="" type="checkbox"/> Drafts in <div>Drafts</div>

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.

From Address

This field allows you to specify the `From:` address for the messages you send using the WebUser Interface. By default, this address is set to the name of your Account on this Server.

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		Auto Wrap: default (Yes) ▼	Text columns: 72 ▼
Reply Header:	On ^T^N ^F wrote:	^T: time, ^F: sender, ^N: new line	
Reply Quoting:	>		

Reply Header

This string is added in front of the quoted text of the original message. It 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)

Reply Quoting

Each line of the copied original message text is prefixed with this string.

Note: If you set this option to an empty string, the original message text will not be included into a reply message.

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.

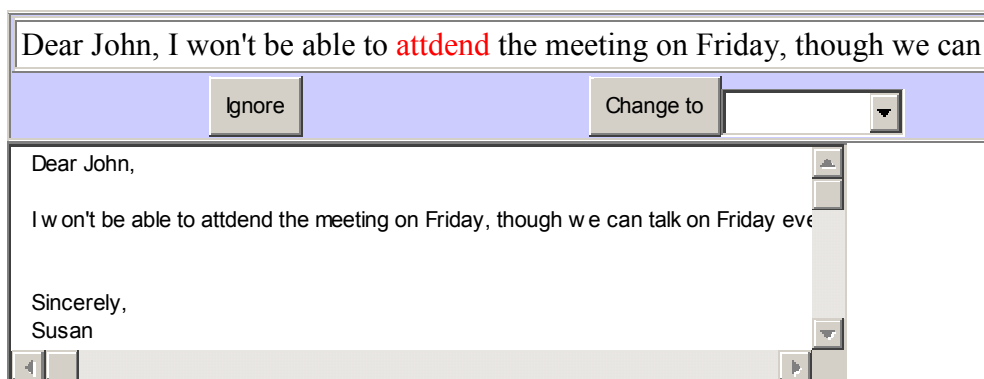
Note: you should attach files after all other message fields are filled. If you click any button (for example, a button that opens the Address Book), the file selection will be cleared and you will have to select the files again.

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.

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:



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.

The screenshot shows a web interface for an address book. At the top, there is a 'Display' button, a dropdown menu currently showing 'SharedContacts', a 'Filter:' label, a text input field containing 'Smith', and a 'Close' button. Below this, on the left side, are buttons for 'To', 'Cc', 'Bcc', 'Delete', and 'Add New'. The 'Add New' button is positioned above a text input field. The main area of the interface displays a list of contacts: 'John S.Smith <j.smith@company.dom>' and 'Thomas Gates <thomas@smithtoyscorp.dom>'. The interface has a light gray background with a darker gray header and footer area.

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.



WebUser Interface: Contacts

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:

Create	Address Book ▼	New Contacts
--------	----------------	--------------

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 Email data displayed:

Display	10 ▼	Filter:		Search:		Mark All
	Name	Email				
<input type="checkbox"/>	John Smith	j.smith@company.dom				
<input type="checkbox"/>	Colt, Susan	susan@coltfamily.dom				
<input type="checkbox"/>	Network Group	[Network engineers in the South]				
New Contact New Contact Group		Copy to... Move to...	Flagged: <input type="checkbox"/> Set Clear			
		C1 ▼	Delete			
		Folder Management				

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
4153837164@domain.dom
no SIP addresses
-----
8002624722@domain.dom
you have answered
transferring
Ringing
```

The Make Call window shows the progress log of your calls.

The Server waits for 15 seconds before cancelling 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:

The screenshot shows a web interface for editing a contact group. At the top, there is a text input field containing "Jim Gray <jim@noc.dom>" and a button labeled "Add New" to its left. Below this, on the left side, is a label "Members:" and a button labeled "Delete". To the right of the "Members:" label is a large text area containing the following text: "John Smith <j.smith@company.dom>", "<operator@company.dom>", and "Tech Support <tech.Support@company.dom>".

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:

The screenshot shows a web-based interface for managing contacts. At the top, there's a header bar with a 'Display' button, a dropdown menu currently showing 'Contacts', a 'Filter:' text input, and a 'Close' button. Below this, on the left side, are several action buttons: 'To', 'Cc', 'Bcc', 'Delete', and 'Add New'. The central part of the interface is a list of contact entries. The first entry is 'John Smith <j.smith@company.dom>'. The second entry, '[@] Jim Gray <jim@noc.dom>', is selected and highlighted with a blue background. The third entry is '[Netw ork engineers in the South]'. At the bottom of the panel, the 'Add New' button is positioned next to a text input field that contains the text 'Susan Colt <susan@coltfamily.dom>'.

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 sim-

ple, 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:

DataSet Address Books	
Name	
<input type="text" value="addressbook"/>	
<input type="text" value="RepliedAddress"/>	
<input type="text" value="BlockedSender"/>	
<input type="text"/>	

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:

Directory Address Books		
Name	Search Base	
My Domain	\$domain\$	
ACME Contact	o=ACME,c=US	
Entire Directory	top	

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:

Main Contacts Mailbox:	<input type="text" value="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:

A horizontal bar with a light gray background. On the left, the text "Main Address Book:" is displayed. To its right is a white rectangular box containing the word "Contacts". To the right of this box is a small downward-pointing arrow icon, indicating a dropdown menu.

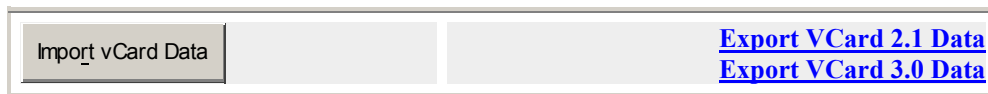
The Address Book panel displays the *Current Address Book*. When you login, the Main Address Book is used as the Current Address Book. When you select a different Address Book in that panel, it becomes the Current Address Book.

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:

A horizontal bar with a light gray background. On the left, there is a button labeled "Import vCard Data". To the right of this button is a large, empty rectangular box. Further to the right, there are two blue hyperlinks stacked vertically: "Export VCard 2.1 Data" and "Export VCard 3.0 Data".

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 calendaring data in the old (2.1) or new (3.0) vCard format, click one of the Export vCard Data links.



WebUser Interface: Calendar

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 WebUser Interface Calendaring functions are available only if the WebCal Service is enabled in the Account settings and in the Account Domain settings.

The Calendar information can be exported as a file in the VCALENDAR format.

The Calendar information can be [accessed via HTTP](#) so groupware applications can subscribe to it.

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 [Personal File Site](#). 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:



The image shows a horizontal bar representing a 'Create' pop-up menu. On the left is a button labeled 'Create'. To its right is a dropdown menu currently displaying 'Calendar' with a small downward-pointing arrow to its right. Further right is a text input field containing the text 'FamilyEvents'.

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:

<div><div><div>△</div><div><=<</div></div><div><div>>=></div><div></div></div><div><div>(-)</div><div></div></div><div><div>(+)</div><div></div></div><div><div><=></div><div></div></div><div><div>>=></div><div></div></div></div>												
2003		8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	
Fri	May 23				Group Meeting			Presentation				
Sat	May 24											
(-) Sun	May 25											
(+) Mon	May 26	Out of Office			Group Meeting		Conf Call w/ACME					
Tue	May 27		ACME Meeting									
Wed	May 28				Group Meeting							
Thu	May 29											
<div><div>▽</div><div>↓</div></div> <div><div>Folder Management</div><div>View As Folder</div></div>												

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:

<=< May, 2003 >=>						
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	31	

Use the arrow links to switch to the previous month or to the next month.

Click the / element to "flip" the table so each day is displayed as a vertical column:

<=< (-) (+) >=>							
2003	Sun, May 25	Mon, May 26	Tue, May 27	Wed, May 28	Thu, May 29	Fri, May 30	Sat, May 31
7:00			PR meeting				
8:00							
9:00		Out of the Office					
10:00				Group meeting		Group meeting	
11:00						Conf Call [X]	
12:00							
13:00							
14:00							
15:00		Prep. to Tue PR Meeting					
16:00							
	Folder Management				View As Folder		

The "All-Day" Events are displayed at the beginning of the day row or column.

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:

When:	Starts:	Mon 26-May-03 ▾	12:00 ▾	<input type="checkbox"/> All-Day Event
	Duration:	30 minutes ▾	Show Availability	
		Add Recurrence	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:

Priority:	Normal ▾
Status:	BUSY ▾
<input type="checkbox"/>	Private Item
<input checked="" type="checkbox"/>	Send Requests

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 Summary of the Event. The Summary 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:

To:	Joe Doe <joe@company1.com>	ISO-8859-1
Summary:	Preliminary PR meeting	
Optional:	Susan <susan@company2.com>	
Inform:		
Location:	Little Town, H&R Hotel conf. room	

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	Status	11:00	11:30	12:00	12:30	13:00	13:30	14:00	14:30	15:00	15:30
	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:	1	day(s)		Remove Recurrence
End by:	Never			

- Every week or every *N*th week, on specified weekdays. Use the Weekly frequency:

Every:	<input type="text" value="1"/> week(s)	Remove Recurrence
On:	<input checked="" type="checkbox"/> Mon <input checked="" type="checkbox"/> Tue <input checked="" type="checkbox"/> Wed <input type="checkbox"/> Thu <input checked="" type="checkbox"/> Fri <input type="checkbox"/> Sat <input type="checkbox"/> Sun	
End by:	<input type="text" value="Never"/> ▼	

- Every month or every *N*th month, on the specified day of month. Use the Monthly by Day frequency:

Every:	<input type="text" value="1"/> month(s) on day <input type="text" value="17"/>	Remove Recurrence
End by:	<input type="text" value="Never"/> ▼	

- Every month or every *N*th month, on the specified week and weekdays of month. Use the Monthly by WeekDay frequency:

Every:	<input type="text" value="1"/> month(s) on <input type="text" value="last"/> ▼	Remove Recurrence
	<input checked="" type="checkbox"/> Mon <input type="checkbox"/> Tue <input type="checkbox"/> Wed <input type="checkbox"/> Thu <input checked="" type="checkbox"/> Fri <input type="checkbox"/> Sat <input type="checkbox"/> Sun	
End by:	<input type="text" value="Never"/> ▼	

- Every year or every *N*th year, on the specified day of year. Use the Yearly by Day frequency:

Every:	<input type="text" value="1"/>	year(s) on	<input type="text" value="19"/>	of	<input type="text" value="Feb"/>	<input type="button" value="Remove Recurrence"/>
End by:	<input type="text" value="Never"/>					

- Every year, on the specified weekdays of specified week. Use the Yearly by WeekDay frequency:

Every:	<input type="text" value="May"/>	on	<input type="text" value="second"/>	<input type="button" value="Remove Recurrence"/>		
<input type="checkbox"/> Mon	<input checked="" type="checkbox"/> Tue	<input type="checkbox"/> Wed	<input type="checkbox"/> Thu	<input type="checkbox"/> Fri	<input type="checkbox"/> Sat	<input type="checkbox"/> Sun
End by:	<input type="text" value="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:

<input type="button" value="Accept"/>	<input type="button" value="Decline"/>	<input type="button" value="Tentative"/>
<div><div></div><div></div><div></div></div>		

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.

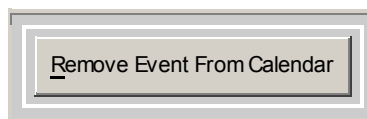
Reconfirming and Declining accepted Requests

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:

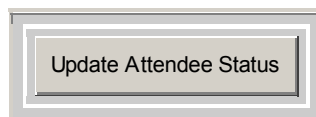


The Cancellation message is sent to all Event attendees and the Event is removed from your Calendaring folder.

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.

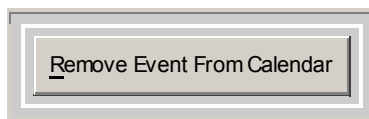
Processing Event Replies

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:

Main 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):

Work Week	<input type="text" value="default(08:00)"/>	<input type="text" value="default(17:00)"/>
Starts at:	<input type="text" value="default(Mon)"/>	<input type="text" value="Default"/>
	<input type="text" value="Mon"/>	<input type="text" value="Tue"/>
	<input type="text" value="Wed"/>	<input type="text" value="Thu"/>
	<input type="text" value="Fri"/>	

The Calendar View panel allows you to customize the Calendar browser:

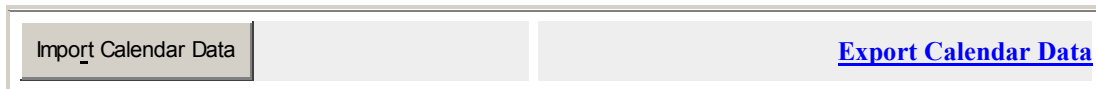
Calendar View	
Days to Display:	<input type="text" value="default(7)"/>
Time to Display:	<input type="text" value="default(10 hours)"/>
Time Slice:	<input type="text" value="default(60 minutes)"/>
Day by Day:	<input type="text" value="default(YES)"/>

When 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:

Free/Busy Publishing
Days to Publish: <input type="text" value="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:



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.

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	authenticated

Action	Parameters
Accept Request	
Reject with	Only Calendaring Event Requests are acc

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 this 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 Event Request is found, the Accept Request Rule action opens the Main Calendar mailbox and parses it, building the Account Calendar. It then checks that the requested event does not conflict with any other Event in the Calendar. If a conflicting Event is found, a negative Event Reply is sent to the Event 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](#) is updated. The incoming message is discarded, and Rule processing stops.

You can specify the `[ignore-conflicts]` string in the Rule action parameter field. In this case the Accept Request Rule action will not check for conflicting events.

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	authenticated
Return-Path	in	ceo@mycompany.com,cto@mycompany.com

Action	Parameters
Accept Request	[force]
Reject with	Only Calendaring Event Requests are accepted

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.



WebUser Interface: Tasks

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 WebUser Interface Tasks functions are available only if the WebCal Service is enabled in the Account settings and in the Account Domain settings.

The Tasks/ToDo information can be exported as a file in the VCALENDAR format.

The Tasks/ToDo information can be [accessed via HTTP](#) so groupware applications can subscribe to it.

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:

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:

△	◀◀											⋮
	2003	6 Jul	7 Jul	8 Jul	9 Jul	10 Jul	11 Jul	12 Jul	13 Jul	14 Jul	15 Jul	
(-)	1			Urgent: Prepare slides for ACME meeting (10%)								
(+)	2			Create the conference report (0%)								
	3			Read the ACME product docs (100%)								
▽	Hide Completed				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 ▾ %
<input type="checkbox"/>	Private Item
<input checked="" type="checkbox"/>	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 Summary. The Summary 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:	Joe Doe <joe@company1.com>
Summary:	Prepare slides for the ACME meeting

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.

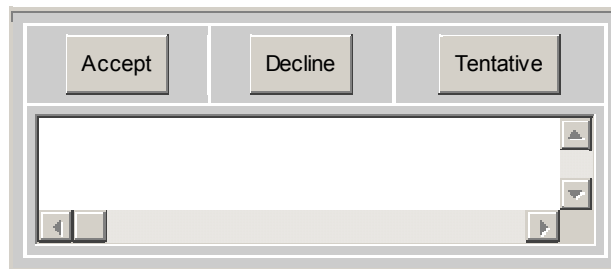
The Task composing page displays a list of those to whom the Task is assigned, along with their confirmation status data. When you receive replies to your Task Assignment request (see below), the confirmation status data is updated. You can also set the status manually if the person has replied with 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).

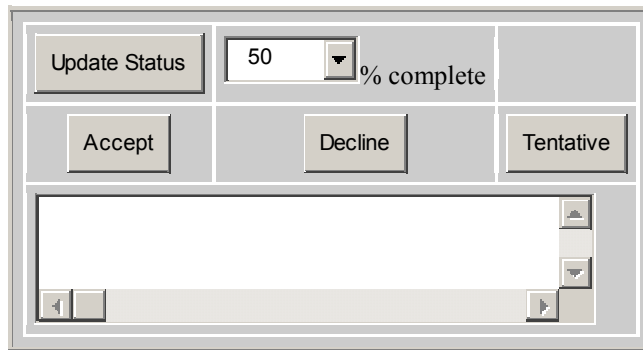
If you click the Decline button, a negative reply is sent back to the Tasks organizer, and the Task is not stored in your Main Tasks folder.

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.



The image shows a task management interface. At the top, there is a button labeled "Update Status" next to a text input field containing "50" and a dropdown arrow, followed by the text "% complete". Below this, there are three buttons: "Accept", "Decline", and "Tentative". At the bottom, there is a large rectangular area with a list of tasks, each represented by a small icon and a text label. The list is currently empty.

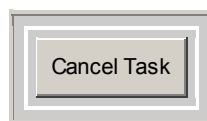
Click the Accept or Tentative button to send a positive response to the Task organizer.

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:



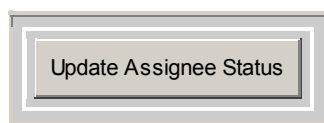
A Cancellation message is sent to those whom you have assigned the Task to, the Task is removed from your Tasks folder.

You can open an existing Task and remove some assignees. When you store the Task, a Cancellation message is

sent to all removed assignees.

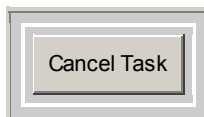
Processing Task Assignment Replies

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:

Main Tasks Folder: <input type="text"/>

The Tasks View panel allows you to customize the Tasks browser:

Tasks View	
Days to Display: <input type="text" value="default(7)"/>	Tasks to Display: <input type="text" value="default(20)"/>

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:

<input type="button" value="Import Tasks Data"/>	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.



WebUser Interface: Notes

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

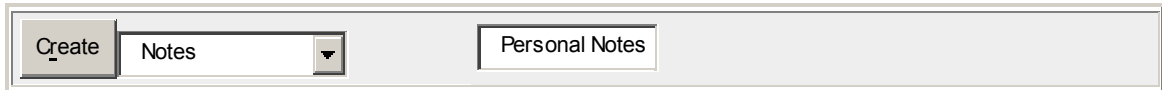
Notes mailboxes ("folders") can be created in your Account using the WebUser Interface or a MAPI client application (such as Microsoft Outlook). These mailboxes appear on the [Mailboxes](#) page. Click on a Notes-type mailbox name to open the Notes list.

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:

A screenshot of a web form for creating a Notes mailbox. It features a 'Create' button, a dropdown menu currently showing 'Notes', and a text input field containing 'Personal 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:

A screenshot of a text input field with the label 'Notes Mailbox:' positioned to its right.



WebUser Interface: Secure Mail (S/MIME)

Secure Mail (S/MIME) 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 possesses 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 algorithms 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 possesses 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 you will not be able to decrypt any message encrypted with your old Public Key. Neither your System Administrator nor Stalker Software 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:

Import Key and Certificate	
PFX File:	
File Password:	<input type="text"/>
<input type="button" value="Import File Data"/>	

Secure Mail Password:	<input type="text"/>
Verify Secure Mail Password:	<input type="text"/>

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:

Generate Key and Certificate	
<input type="button" value="Generate"/>	

Secure Mail Password:	<input type="text"/>
Verify Secure Mail Password:	<input type="text"/>

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.

Modify Secure Mail Password	Remove Key and Certificate
New Password: <input type="text"/>	Saved PFX File: <input type="text"/>
Verify Password: <input type="text"/>	File Password: <input type="text"/>
<input type="button" value="Modify Password"/>	<input type="button" value="Compare with File and Delete"/>
Export Key and Certificate	

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 PFX file	
File Password:	<input type="text"/>
Verify File Password:	<input type="text"/>
<input type="button" value="Export"/>	

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

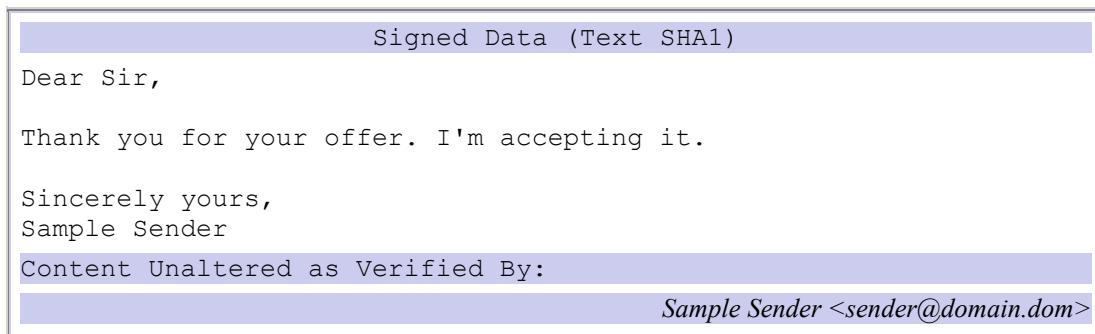
When the Private Key is placed into the Account Settings, it is *activated*. The WebUser Interface automatically decrypts all messages encrypted with your Certificate/Public Key, and you can send encrypted and signed messages. In order to protect your sensitive information, your Private Key is automatically 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:

Secure Mail is Locked	
SMIME Password:	<input type="text"/>
<input type="button" value="Unlock Secure Mail"/>	

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:



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.

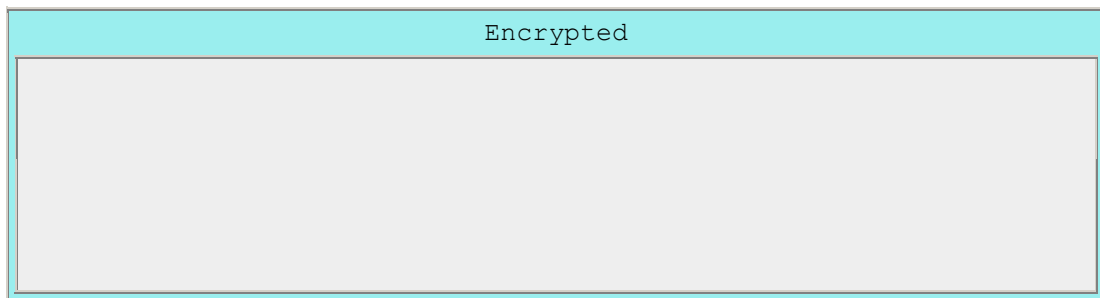
If your Private Key is unlocked, you will see the Send Encrypted checkbox on your Compose page. Select this checkbox to encrypt your message. If you send a message with attachments, the entire content of your message, including all attachments, will be encrypted with the recipients Public Keys (taken 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.

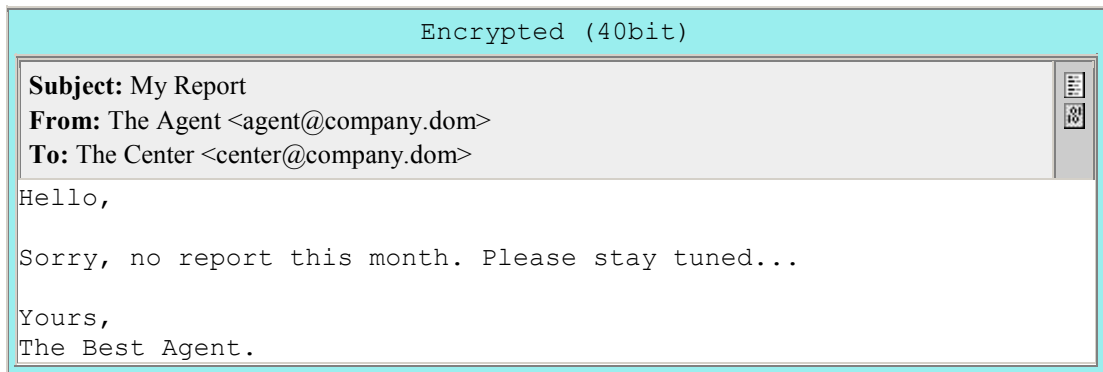
Use the Encryption Method WebUser setting to specify the encryption "cipher".

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.



PBX Services

The CommuniGate Pro Server provides Real-Time Signalling services. Some of these services were implemented in the legacy telephony using "phone switching stations" also called *PBX* (Private Branch Exchange), thus the name of this CommuniGate Pro component.











If the PBX [Service](#) is enabled for your Account, you can use the [WebUser Interface](#) to manage your PBX service settings.

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.

The CommuniGate Pro PBX functionality requires a Groupware-type License Key. Without that License Key, the trial-version reminder message is periodically played for all media sessions terminated within CommuniGate Pro.

Call Control

The Call Control settings allow you to specify how your incoming calls are handled. Use the [WebUser Interface](#) to log into your Account, and open the Call Control subsection of the Settings section:

Incoming Calls			
	If Busy:	redirect	 default (voicemail)  frontdesk@example.com
After Ringing for	10 seconds	redirect	 default ()  secretary@example.com
If No Answer for	default (15 seconds)	default (start)	 default (voicemail) 
	On Failure:	start	 default ()  voicemail
	On Self-Call:	default (start)	 default (service) 

If Busy

This tells the system what to do when it gets a "Busy" or "Not Available"/"Do not Disturb" response from your device(s). You can redirect calls to a different account (hosted on the same CommuniGate Pro Server or on any other system in the world), or you can start an application, such as the [voice-mail_application](#).

After Ringing

This setting tell the system to "fork" your call if it was not answered during the specified period of time. While the system continues to ring all your registered devices, it sends (redirects) the call to a different user (hosted on the same CommuniGate Pro Server or on any other system in the world), or to an application.

For example, you may want your incoming calls to ring your secretary phone(s) if you do not answer the phone within a few seconds.

If no Answer

This setting tells the system what to do if the call is not answered within the specified period of time. If you do not have any devices registered, or zero time is specified in this setting, the action is taken immediately:

you can tell the system to redirect the call to some other destination, or to start an application (such as the [voicemail application](#)) and redirect the call to that application.

On Failure

This setting tells the system what to do when a call cannot be completed because your device(s) reported some errors.

On Self-Call

This setting tells the system what to do when you call your own Account. Usually, these calls are directed to the [service application](#).

If your Account has no registration (i.e. none of your devices is active), then the After Ringing action is taken immediately (if enabled).

If the After Ringing option is not enabled, and your Account has no registration, then the If No Answer action is taken immediately (if enabled).

When specifying a Redirect action, you can specify a redirection address as:

- a simple name or 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>*)

If the specified address contains the star (*) symbols, the first star symbol is substituted with your Account name.

You can specified several addresses if you separate them using the comma (,) symbols.

PINs (Numeric Passwords)	
Service Access PIN:	<input type="text" value="*****"/>
Conference PIN:	<input type="text" value="*****"/>

Service Access PIN

Enter a number into this field. You will use this number (PIN) to access your Account [services](#) 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 [Personal Conferencing](#) facility.

Voice Mail

Your Account is usually configured to start the `voicemail` application when you are busy, not available, or not answering.

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](#).

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 your 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 clicking them.

You can use the `voicemail` application to start the Services application for your account when you cannot call from your own SIP phone (for example, when you are calling from a PSTN phone via a gateway).

While listening to the greeting and/or the menu options, press the * (star) symbol twice. The application will ask you to enter your Access PIN. If the number is correct, the application switches you to the Services application menu.

Services

Your Account is usually configured to start 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.

You can also invoke the `service` application by calling your Account from any device and using the voice-mail application [Access PIN](#) feature.

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.

The stock `service` application supports the following function codes:

- [Inbox](#) management.
- [voicemail greeting](#) management.
- [custom greetings](#) management.
- [conference](#) start.
- [call park](#).
- silent [call park](#).
- [parked call pick up](#).
- [missed call pick up](#).
- [last call return](#).

Mailbox Management

The [service](#) application 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, and to mark your messages for deletion.

Voicemail Greeting

The [voicemail](#) application checks if your Account [File Site](#) 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 [File Site](#) modification methods - FTP, WebUser Interface, etc.

You can create, modify, or remove your personal greeting by starting your [service](#) application 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 [service application](#) 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 [File Site](#) 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 Site](#) modification methods - FTP, WebUser Interface, etc.

Call Park and 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 `voice-mail` 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 [service_application](#) 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 connect you to some number.

Dial ***40** or connect to your [service application](#) 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 numbers. Use the [WebUser Interface](#) 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 [service application](#) and enter the **4N** code, where *N* is the speed-dial digit code. The system will dial that number on behalf of your Account.

Conference Host

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".

PBX 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 SIP servers, PSTN users calling via gateways, etc.)

The PBX Center allows your Domain users to collaborate on call processing.

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 SIP address" of your organization.

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 SIP 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 SIP phones.

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.

When the PBX Center is called directly (using the pbx name or the 200 Alias), it starts its [Auto-Attendant](#) function.

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".

Auto-Attendant

An auto-attendant is an application that 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. Open the Real-Time Preferences page of the PBX Center Account and follow the Advanced link to open the Auto-Attendant Preferences:

Auto-Attendant					
Language Menu:	english	french	japanese		
Department Menu:	sales	techsupport	operator	marketing	
Directory Prefix:	2				
Directory Digits:	3				

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 Auto-Attendant 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 `forname.wav`, where *name* is the "department name".

The stock Environment contains media files for the following "department names": `administration`, `confcenter`, `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 SIP devices that have digital keypad only, Accounts should have digital [Aliases](#). 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 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.

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.

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, 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 then 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.



Miscellaneous

This chapter describes various CommuniGate Pro 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 header 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: header 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 [-iInv] [-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.

`-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] [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.

-f *from-address*

Use the *from-address* as the message From: address. If this parameter is not specified, the current user name is used.

-F *sender-name*

Set the full name of the sender.

-V *envid*

The Evelope ID of the message.

-O*parameter*

-oparameter

Option ignored.

addresses

The destination addresses (without the `-t` option) or the addresses to be excluded from 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.

A

- ACAP 7, 905
- ACAP access 759
- ACAP module 575, 645
- AcceptBridge 1139
- Access 66, 575
- Access Control Lists 597
- access right 61
- Access Right Records 693
- Access to Accounts 576
- Access to Calendar and Tasks data 652
- Access to Personal File Sites 652
- Access to the WebUser Interface 651
- Accessing the Server MIB 679
- Account 5
 - Settings 713
- Account Access 604
- Account Aliases 274, 315
- Account Data 355
- Account Files Location 357
- Account Import file 98
- Account Information 5
- Account Log 279
- Account Mailboxes 428
- Account Password Attacks 195
- Account Passwords 188
- Account Services 648
- Account Settings 5, 172, 318
- Account Storage 5
- Account Template 60, 320
- Accounts 94, 272, 305
- Adding a Backend Server 784
- Adding a Frontend Server 784
- Adding Required Headers 1360
- Adding Servers 774
- Adding Subscribers 468
- Additional Mailboxes 590
- Address Book 1295, 1301
- Address Structure 146
- Address Testing 1359
- Administration 9
- Administrator Domain 293
- Advanced Security 7
- Alerts 239, 1243
- Aliases 5, 325
- All-Domain Aliases 164
- Anti-Spam 6
- AORs 481
- Application Configuration Access Protocol 645

Application Environments 1045
Application Model 1049
Applications 921
Archiving 465
Arrays 925
Assigning IP Addresses 764
ATRN 56
ATRN/PROP 55
AttachmentPart 1252
AttachMixer 1142
Attributes Renaming 708
Authentication 486
Authentication Methods 185, 307
Auto-Attendant 1354
Automated Invitation processing 6
Automated Mail Processing 427
Automated Mail Processing Rules 6, 360
Automated Processing 1023
Automated Rules 1023, 1268
Automated Signal Processing Rules 6, 493
Automatic Blacklistings 178
Automatic Encryption 7
Automatic S/MIME Encryption 215
Auto-Restart 394
Auto-Signup 608, 1263
away 488

B

Backend Failover 775
Backend Queues 857
Backend Server 772
Backup Controller 779
Base Directory Structure 114
be-back 488
Blacklisted Addresses 416
Blacklisting Domains by Name 176
Blacklisting Offenders 174
Bounce Processing 455
BreakBridge 1139
Browser Authentication Login 1262
busy 488
Bye 1246

C

Calendar 27, 1229
Calendar component 360
CalendarPart 1255
Call Control 1345
Call Control settings 1345

Call Detail Records 491
Call Park 1350
call park 1349
Call Park Center 1357
Call Pick up 1350
Calls 507
Central Directory 68, 665
Certificate 200, 204, 309
Certificate Authentication 193
Certificate methods 649
Certificates 1336
CG/PL 7, 983
 Account Data 1006
 Addresses and URIs 1005
 Alternative Forms 994
 Arrays 1003
 Built-in Procedures and Functions 999
 Code Sections 995
 Communications 1011
 Data Conversion 1004
 Data Model 984
 Datablocks 1003
 Dictionaries 1004
 Environment 1005
 Events 1016
 Expressions 985
 Formal Syntax 1020
 Functions 998
 IP Addresses 1003
 Language Elements 983
 Lexemes 984
 Mailbox handles 1008
 Meeting 1017
 Multitasking 1015
 Numbers 1001
 Operators 991
 Procedures 997
 Queues 1019
 Services 1010
 Spawning 1015
 Strings 999
 TimeStamps 1001
 Variables 985
CG/PL Applications 1128
CG/PL language 1128
CG/PL Programs 497
CGI programs 197
CGPL files 1045
Channels 399, 432

CHAP 673
CLI 929
 Access Rights Administration 977
 Account Administration 931
 Alert Administration 954
 Domain Administration 940
 Forwarder Administration 939
 Group Administration 938
 Mailbox Administration 949
 Mailing Lists Administration 958
 Miscellaneous Commands 979
 Monitoring 976
 Real-Time Application Administration 969
 Server Settings 972
 Statistics 977
 Syntax 930
 Web Interface Integration 968
 Web Skins Administration 963
CLI API command 122
CLI commands 930
CLI interface 97
CLI/API 8
CLI/API interface 647
Client Authentication 663
Client Certificate-based Secure Authentication methods 7
Client Certificates 214
Client Domains by Name 168
Client IP Addresses 168, 250
Cluster 757
Cluster Access 905
Cluster Communication 762
Cluster Configuration 766
Cluster Controller 778
Cluster File System 812
Cluster File Systems 777, 780
Cluster Frontends 831
Cluster Network 761
Cluster Of Clusters 768
Cluster of Clusters 8
Cluster OSes 780
Cluster Real Time 895
Cluster Server Configuration 761
Cluster Setup 184
Cluster Storage 787
Cluster Transfer 831
Cluster Types 758
Cluster WebAdmin page 784
Cluster-wide Directory Integration Settings 785
Cluster-wide Lawful Interception settings 785

Cluster-wide Network Settings 785
Cluster-wide Protection Settings 785
Cluster-wide Real Time Applications 785
Cluster-wide Router Table 785
Cluster-wide Routing Table 165
Cluster-Wide Rules 1030
Cluster-wide Rules 785
Cluster-wide WebSkins 785
Code Components 1213
Command Line Interface 929
Command Line Options 119
Common Gateway Interface 658
Common Gateway Interface (CGI) 649
CommuniGate Password 308
CommuniGate Passwords 188
CommuniGate Pro
 Directory 707
 Domains 707
CommuniGate Pro Cluster 197
CommuniGate Pro Domain 1046
CommuniGate Pro Domains 57, 301
CommuniGate Pro Server supports 185, 186, 188, 193
CommuniGate Pro String format 1039
CommuniGate® Pro 5
Compose 1239
Compose Limit 606
Composing Service Texts 445
Concepts 921
Concurrent Requests 254
Conference Center 1355
Conference Host 1352
conference start 1349
Configuring 217, 243
Configuring Backend Servers 782
Configuring Mailing Lists 442
Configuring SNMP Agent 677
Configuring the ACAP module 646
Configuring the FTP module 638
Configuring the HTTP modules 654
Configuring the IMAP module 596
Configuring the LDAP module 662
Configuring the LIST module 440
Configuring the MAPI Connector 616
Configuring the PIPE module 478
Configuring the POP module 588
Configuring the PWD module 670
Configuring the RADIUS Module 674
Configuring the RPOP module 432
Configuring the Signal Component 484

Configuring the SMTP module 169, 398
Configuring the TFTP module 642
Confirmation Requests 449
Contacts 1229
Contacts Mailboxes 1297
Control Lists 340
Controller Backup 779
Converting Passwords 97
Copying All Mailboxes 106
Copying Mailboxes 102, 103
CPL Scripts 497
Create Domain 279
Creating a Mail Profile 616
Creating Calendar Mailboxes 1308
Creating Contacts Mailboxes 1297
Creating Mailboxes 347
Creating Mailing Lists 441
Creating Notes Mailboxes 1332
Creating Skins 1183
Creating Tasks Mailboxes 1324
Custom Account Settings 714
Custom Greeting 1350
custom greetings management 1349
Custom Settings 741

D

Data Access Services 7
Data Formats 923
DataBlocks 924
DataSet Address Books 1302
Date and Time 68
Default IP Address 400
Default Records 162
Default Schema 701
default Skins 785
default.html 353, 1215, 1219
DeliveryReportPart 1252
Dequeuing 366
Dequeuer component 360
DetachMixer 1142
Dialog 1143
Dialup 265
Dial-up Client 412
Dictionaries 926
DIGEST/INDEX Mode Distribution 461
Digesting and Archiving 461
DIGEST-MD5 673
Direct Mailbox Addressing 428
Direct Mapping 302

- Directory 5, 67, 683, 684
 - Integration 707
 - Integration in a Cluster 740
 - Routing 753
 - setting 709
- Directory Address Book 1303
- Directory Browser 697
- Directory Digits 1355
- Directory Integration 68, 285, 707
- Directory Manager 683
- Directory Prefix 1355
- Directory Schema 701
- Directory Storage Units 686
- Directory-based Domains 741
- Displaying the Domain List 278
- DispositionReportPart 1253
- Distributed Domains 8
- DNS A-Record 281
- DNS MX records 53
- DNS MX-Record 281
- DNS records 147
- Domain 506
- Domain Administration 124
- domain administration rights 114
- Domain Administrator access right 124, 650
- Domain Administrator Access Rights 125
- Domain Administrators 650
- Domain Aliases 284, 302
- Domain File Directories 295
- Domain Files 355
- Domain Limits 283
- Domain Name Resolver 130, 254
- Domain Name System 277
- Domain Objects 270
- Domain PKI Settings 201
- Domain Rules 293
- Domain S/MIME Settings 215
- Domain Security Settings 293, 295
- Domain Settings 172
- Domain TLS certificates 261
- Domain WebAdmin 649
- Domains 269, 277
 - Subtree 709
- Domains Subtree 776
- Domain-Wide Rules 1030
- DTMF 1135
- DTMF symbol 1129
- Dynamic Cluster 777, 787, 1046
- Dynamic Clusters 8

E

- EAP 673
- Edge Services 536
- Enabled Services 310
- Enabling Auto-Signup 289
- Enabling Mailbox Sharing 622
- Enabling Services 281
- EncryptedPart 1253
- Enqueueing 364
- Enqueuer component 360
- Environment Files Hierarchy 1046
- Error Pages 1221
- ESMTP 6
- ETRN 54, 56
- Event Handler component 360
- Event Handlers 8
- Event Packages 487
- Event packages 6
- Expressions 1188
- External Applications 475
- External Authentication 6, 108, 193, 309
- External Authenticator 1035
- External CDR Processor 1043
- External Filter 393, 395
- External Filters 391, 392
- External Helper 675
- External Helper Programs 132
- External Helper Routing 159
- External INBOX 96
- External Mailbox 583
- External Mailboxes 585
- External Message Filter 1039
- External Password 188
- External Program Delivery 6
- External RADIUS Helpers 1040

F

- Far-End NAT Traversal 525
- Features 8
- Features and Standards 11
- FEED Mode Distribution 458
- File Access 913
- File Servers 777
- File Sites 349
- File Store 6
- File Systems 787
- File Transfer Protocol 637
- Filtering 138
- Filtering Mail 181

Foldering Method 297, 299
Foreign Mailbox Access 583
Foreign Queue Processing 479
Forking 484
Forwarders 5, 273, 333
Free/Busy Information 623
freebusy.vfb 353
freebusy.wssp 354
Frontend Server 772
Frontend Servers 759
FTP 7, 26, 758, 767, 913
FTP module 575, 637
FTP-based Management 353

G

Gateway Name 506
General Settings 115
Generic Code Components 1247
Generic E-mail 15
Group Settings 328
Groups 5, 273, 327
Groupware Information 6
GSSAPI 7

H

Helper Applications 1033
Helper Protocol 1033, 1035, 1039
History 35
How To 51
HTML messages 57
HTML-based Management 351
HTTP 7, 913
HTTP Access 350
HTTP Modules 649
HTTP modules 647
HTTP requests 61
HTTP servers 1177
HTTP-based Management 352
HyperThreading 226

I

iCalendar 6
Identity Management 5
Ignore 325
ImagePart 1252
IMAP 905
IMAP access 758
IMAP Extensions 600
IMAP Implementation 600

IMAP Module 595
IMAP module 360, 575
IMAP4 7
IMAP4 Clients 223
Importing 699
Importing Account Information 324
Inactivity Time-Out 606
INBOX 306
index.wssp 354
Initial Configuration 88
in-meeting 488
Input 1129
Installation 71
Installing on a BeOS System 87
Installing on a BSDI BSD/OS System 78
Installing on a FreeBSD System 76
Installing on a Linux System 72
Installing on a MacOS Rhapsody System 87
Installing on a MacOS X (Darwin) System 75
Installing on a MS Windows 200x/XP/NT/9x/ME System 74
Installing on a NetBSD System 76
Installing on a QNX System 85
Installing on a Sun Solaris System 71
Installing on a Tru64 (Digital Unix) System 80
Installing on an AIX System 78
Installing on an HP/UX System 79
Installing on an IBM OS/2 System 86
Installing on an IBM OS/400 System 83
Installing on an OpenBSD System 77
Installing on an OpenVMS System 84
Installing on an SCO OpenServer System 82
Installing on an SCO UnixWare System 81
Installing on an SGI IRIX System 81
Installing the MAPI Connector 615
Integrating Regular Domains 716
integration 707
Interface Login 1261
Interfaces 905, 913
International 14
Internet Message Access Protocol 595
Internet Standards 8
IP Address 925
IP Addresses 925
IP Aliasing 781
IP-PBX 983
IsBreakBridgeEvent 1139
IsStartBridgeEvent 1139

K

Kerberos 186, 309, 649
Kerberos Authentication 190
Kernel 11

L

LAN Addresses 249
LAN IP Addresses 250
Language files 1178
Large Domains 298
Last call return 1351
Lawful Interception 8, 217
Lawful Interception component 360
LDAP 5, 67, 768, 913
LDAP module 647, 661
LDAP Provisioning 98, 666
LDAP servers 683
LDAP-based Provisioning 8, 725
Legacy Mail Emulation 1360
Legacy Mailboxes 96
Lightweight Directory Access Protocol 661
Limiting Connections 263
LIST module 360, 439
List.html 1219
listener 406
Listeners 259, 766
LMTP 6, 18
Load Balancer 765, 831
Local Delivery 55, 844
Local Delivery Module 63
Local Delivery module 359, 421
Local Directory 689
local IP address 260
Local Nodes 489
Local Units 689
Local Users 96
Locale 94
Locked Mailboxes 346
Login Referrals 598
Logs 68

M

Mail Distribution 436
Mail Domain Name 146
Mail Receiving 265
Mail Storage 311
Mail Store 6
Mail To All Accounts 287, 288
Mail Transfer 6

Mailbox 1226
Mailbox Access Rights 59
Mailbox Aliases 348
mailbox aliases 60
Mailbox Classes 346
Mailbox component 1247
Mailbox Formats 343
Mailbox formats 6
Mailbox Management 1349
Mailbox Names 337
Mailbox Sharing 583
Mailbox Subscription 59, 347
Mailboxes 59, 274, 311, 337, 1224
Mailboxes Setting 60
MailboxSettings 1241
MailDir Mailbox 306
Mailing List 6
Mailing Lists 19, 273, 317, 439, 608
Main Calendar Mailbox 1307
Main Contacts Mailbox 1297
Main Domain 67
Main Domain Name 64, 94, 115
Main Notes Mailbox 1331
Main Server Domain 301
Main Tasks Mailbox 1323
Managing Environments 1047
Managing Skins 1183
Many Concurrent Clients 222
MAPI 7, 613, 905
MAPI client 360
MAPI Connector 613, 614
MAPI module 575
MAPI provider 613
Mapping 301
Media 1137
Media Formats 1144
Media Proxy Parameters 252
Media Stream Proxy 508
Message 1235
Message component 1250
Message Flags 338
Message Flow Control 423
Message Relaying 414
Message Rules 373
message scanning 1033
Message Waiting Indication 488
Migrating 95
Miscellaneous 1359
Missed Call Pick up 1351

MixerAttached 1143
Modifying Mapping 302
Monitor Access Right 136
Monitoring 679
Monitoring a Message 370
Monitoring a Queue 368
Monitoring IMAP 599
Monitoring SMTP Activity 419
Moving 90, 110
MS-CHAPv1 673
MS-CHAPv2 673
MultiAccess 597
Multi-Domain 5, 648
Multihoming 280, 578, 648
Multi-lingual Skins 7
MultiMailbox 306
Multiple Domains 603
Multi-Server Operation 8
Multi-Socket Listening 259

N

Name 325
NAT Traversal 6, 508
NAT/Firewall 251
NATed Addresses 250
Near-End NAT Traversal 515
Netscape Roaming 357
Network 94, 249
Network File System 796
Network Users 95
New Account 305
New Group 327
New Subscribers 452
Nodes component 490
Non-Standard Ports 419
Notes 1229
Notification 244
Notification Alerts 590, 598
Numbers 924

O

Object Classes 702
objectClass 741
Objects 269
offline 488
online 488
on-phone 488
OS Limitations 228
OS Password 97

OS Passwords 189

OS Tuning 228

out-lunch 488

P

PAP 673

Password 325, 1245

Password Editor page 1266

Password Modification 1266

Password Modification Protocol 669

Password.wssp 1217

PBX Center 1353

PBX Center application 1356

PBX Center functions 1353

PBX Services 1345

Personal Conferencing 1351

Personal File Sites 349, 575, 649, 759

PIPE Module 473

PIPE module 360

PKI 199, 1333

Play 1137

PlayFile 1137

PlayFileInLoop 1137

PlayInLoop 1137

Plugin Interface 6

POP 25, 905

POP Module 587

POP module 360, 575

POP3 7

POP3 access 758

POP3 Clients 222

POP3 server 587

Poppwd 8

Post Office Protocol 587

Posting Alerts 239

Posting Messages 451

Postmaster 63, 64

Postmaster Account 768

Postmaster Password 93

Preferred Language 118

Preferred Time Zone 118

Presence Server 488

Private Folder 351

Private Key 199, 202, 1336

Processing Messages 454

Processing Requests 482

Processing Unknown Names 285

Processors 422

Programming Language 983

Protection 52, 167
Protection Black-List 262
Protocol 502, 504
Proxy 6
Public Info Editor 1267
Public Key 199, 1336
Public Key information 1336
Public Key Infrastructure 199
Public Mailboxes 584, 597
PublicInfo 1245
PWD 913
PWD access 759
PWD module 647, 669
PWD session 930

Q

Queue Foldering 363
Queue Limit 363
Quick Start 93

R

RADIUS 6, 758, 768, 913, 1033
RADIUS Authentication 675
RADIUS module 647, 673, 1040
RBL 175
ReadInput 1129
Real Time Communication 895
Real Time Communications 481
RealName 325, 329
Real-Time 1045
Real-Time Application 482
Real-Time Application Environment 7
Real-Time Application module 1045
Real-Time Applications 360, 983
Real-Time Communications 630
Real-time Node 482
Real-Time Nodes 489
Real-Time Signal control 983
Real-Time Signalling 6
Real-time Task 482
Receiving Messages 405
Recipients/Message 400
Record 1137
Record Attributes 701
Redirect All 389
Redirect-type Response 1257
Registrar Services 486
Reject Automatic Messages 330
RejectBridge 1139

- relay mail 54
- Remote Account Polling 6
- Remote Accounts 437
- Remote administration features 9
- Remote Directory Root 691
- Remote Queue Processing 878
- Remote Storage Unit 690
- Removing Accounts 318
- Removing Domains 294
- Removing Groups 332
- Removing Mailing Lists 444
- Renaming Accounts 317
- Renaming Domains 294
- Renaming Groups 331
- Renaming Mailing Lists 444
- Reply-To to Group 330
- Return Receipts 1359
- RFC822Header 1252
- RFC822Message 1251
- Router 52, 53, 55, 61, 66, 145, 591
- Routing 51, 364, 418, 424, 425, 471, 581, 657
- Routing by IP Addresses 158
- Routing Settings 429
- Routing Table 147
- Routing via Modules 158
- RPOP 767
- RPOP Accounts 1268
- RPOP Module 431
- RPOP module 359
- Rule Actions 379, 496
- Rule Conditions 374, 494
- Rules 56, 325, 373, 483, 1023
 - Actions 1028
 - Conditions 1025
 - Creating 1024
 - Removing 1024
 - Renaming 1024
- Rules Activity 390

S

- S/MIME 7, 214, 1333
- SASL 7
- SASL authentication methods 186
- Scalability 221
- Scripting Elements 1187
- Searching 140
- secondary Domain 64
- Secondary Domains 116, 301, 650
- secondary Domains 280

Sections and Privileges 113
Secure Access 589
Secure Connections 212
Secure Mail 7, 1270, 1333
Secure Sockets 261
Security 13, 185
Security Issues 766
Send AUTH 401
Sending Mail 430
Sending Messages 399
Sending to an Account 485
Server Logs 114, 135
Server OS Integration 292, 308
Server Port 618
Server Root Privilege 123
Server statistics 114
Server WebAdmin 649
Server-side Encryption 632
Server-wide Rule 56
Service 913
Service Calls 486
Services 647, 1348
Serving LAN Clients 267
Serving Large Domains 221
Serving Multiple Domains 577
Serving Regular Files 1181
Session Initiation Protocol 499
Session Requests 1222
Session Time Limit 606
Session-based Processing 1177
Settings 1264
Shared
 Directory 751
shared 576
shared Domain 278
Shared Domains 764, 771, 772
Shared File Systems 787
Shared Mailboxes 59
Shared Processing 785
Shared Settings 785
Shutting Down 122
Signal flow 483
Signal Module 488
Signal Rules 493
Signals 482, 1130
SignedPart 1254
Signup.wssp 1218
silent call park 1349
Simultaneous Access 583

Single OS File Systems 788
SIP 6, 499, 895
SIP Client Settings 503
SIP Devices 560
SIP Module 499
SIP Module server 482
SIP protocol 499
SIP Server Settings 500
SIP signalling 758
SIP/RTP Clients 224
Skin Files 1180
Skin Text Dataset 1181
Skins 1178
SMTP 16, 767
SMTP AUTH method 171
SMTP Delivery 52, 226
SMTP Force AUTH option 408
SMTP mail delivery 758
SMTP mail receiving 758
SMTP Module 397
SMTP module 52, 53, 54, 55, 359
SMTP Options 290
SMTP Receiving 65
SMTP Relaying 831
SMTP Sending 65
SMTP Servers 53
SNMP 8
SNMP module 647
SNMP server 677
Spam Traps 180
Special Addresses 154
Special Features 589
Special files 353
Special Headers 436
Specifying a Time Interval 137
Specifying Account Settings 306
Specifying Events 247
Specifying Rules 1023
Spell Checkers 606
SSL/TLS 7, 65, 212
standard LDAP Directory Schema 715
StartBridge 1138
Starting 392
Static Cluster setup 771
Static Clusters 8, 771
Storage 325
Storage Area Network 811
Storage Management 6
Storage Quota Alerts 241

Storage Systems 787
Storing Mail 428
String format 1039
Strings 923
Structural Elements 1202
Submitted Folder 473
Submitting Messages 361
Subscriber Lists 468
Subscribers List 466
Subscription 1244
Subscription Modes 448
Subscription Processing 446
Subtree panel 709
Support 9
Supported Services 758
Switching Servers 109
Synchronized 707
Syntax Rules 927
System Tuning 224

T

Tasks 1234
TCP Activity Schedule 266
TCP port 113
TCP Ports 253
TCP/IP networks 613
Text Elements 1195
Text Mailbox 306
TFTP 7, 26, 758, 913
TFTP module 641
TFTP server 641
Time Stamps 143, 925
Time Zones 143
Time-out 394
TLS-Certificate 186
Transactions 502, 504
Transfer 359
Transport 500, 503
Trash Management 1269
Trivial File Transfer Protocol 641
Trusted Root Certificate 210

U

UDP Listeners 263
UDP Ports 253
uid.html 1221
Unified Domain-Wide Accounts 426
UnixPassword 325
Unknown Accounts 425

- Unnamed Skin 1213
- Unsubscribe 467
- Upgrading 90
- Upgrading Servers 786
- user Accounts 649
- User Authentication 589, 598
- user authentication 1033
- user domain settings 96
- User Mailboxes 61
- Username 506

V

- Vacation Message 388
- vCard 6
- vCardPart 1256
- Virtual Files 354
- Voice Mail 1348
- Voice Messages 1348
- Voicemail Greeting 1349
- voicemail greeting management 1349
- VoiceXML Applications 1144
- VoiceXML files 1045

W

- WAP/WML Skins 1179
- Web Access 66
- Web Application Code Components 1213
- Web Application module 1177, 1213
- Web User Interface module 360
- Web/FTP site 71
- WebAdmin 63, 767
- WebAdmin Interfac 64
- WebAdmin Interface 113
- WebAdmin interface 8
- WebAdmin Interface Pages 650
- WebAdmin Preferences 128
- WebMail 1259
- WebMail Integration 629
- WebMail module 575
- WebSite 1246
- WebUser 356
- WebUser Authentication Method 197
- WebUser Clients 223
- WebUser Interface 7, 59, 60, 61, 603, 649, 913
 - Access to Mailboxes 1271
 - Access to Mailboxes by Name 1280
 - Address Book 1295
 - Address Books 1301
 - Assigning Tasks 1326

Attaching Files 1293
Automatic Request Processing 1319
Calendar 1307
Calendar Browsing 1308
Calendar Mailboxes 1307
Calendar Settings 1317
Calling the Contact 1299
Canceling an Event and Attendee Removing 1316
Canceling Tasks 1328
Certificate Storage 1337
Certificates 1336
Checking Spelling 1293
Composer Settings 1290
Composing Messages 1289
Contacts 1297
Contacts and Address Book Settings 1304
Contacts Mailbox Browsing 1298
Creating and Editing Contact Group Items 1300
Creating and Editing Contact Items 1299
Creating and Editing Notes 1332
Creating Calendar Events 1311
Creating Tasks 1325
DataSet Address Books 1302
Delivery Status Notification 1294
Digital Signatures 1335
Directory Address Books 1303
Encrypting Incoming Messages 1344
Encrypting Stored Messages 1343
Forwarding Messages 1292
Importing and Exporting Calendar Data 1319
Importing and Exporting Contacts Data 1305
Importing and Exporting Tasks Data 1330
Mailbox Aliases Management 1279
Mailbox Browsing 1273
Mailbox Management 1277
Mailbox Subscription Management 1278
Mailboxes 1271
Message Browsing 1281
Message Copying 1286
Message Disposition Notification 1294
Message Redirecting 1286
Messages 1281
Notes 1331
Notes List Browsing 1332
Notes Mailboxes 1331
Notes Settings 1332
Opening the Composer Page 1289
Private Key 1337
Private Key Activation 1340

Processing Event Replies 1317
Processing Task Assignment Replies 1329
Public Key Infrastructure 1333
Receiving Encrypted Messages 1342
Receiving Signed Messages 1340
Reconfirming and Declining accepted Requests 1316
Recording Certificates 1341
Replying to Meeting Requests 1315
Replying to Messages 1291
Replying to Task Assignment Requests 1326
Secure Mail 1333
Sending Encrypted Messages 1342
Sending Signed Messages 1341
Storing Addresses 1285
Storing and Removing Attachments 1287
Task List Browsing 1324
Tasks 1323
Tasks Mailboxes 1323
Tasks Settings 1329
Updating Task Status 1327
WebUser Interface access 758
WebUser Interface Pages 1260
WebUser Interface Settings 289, 605
WebUser Interfaces 647
WebUser session 197
WebUser Sessions 604
WebUser sessions 67
Windows applications 614
WML browser 1179
WML Skins 1179
WML/IMode Login 1262
Working Offline 625
WSSP elements 1188
WSSP file 1187
WSSP Files 1182
WSSP files 1178
WSSP mechanism 1221
WSSP Scripting 1187

X

XTND XMIT Extension 590