

Sun Java System Messaging Server 6.3 Administration Guide



Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819-4428-15
11 September 2008

Copyright 2008 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more U.S. patents or pending patent applications in the U.S. and in other countries.

U.S. Government Rights – Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, the Solaris logo, the Java Coffee Cup logo, docs.sun.com, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. This product includes software developed by Computing Services at Carnegie Mellon University (<http://www.cmu.edu/computing>).

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Products covered by and information contained in this publication are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical or biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2008 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plusieurs brevets américains ou des applications de brevet en attente aux Etats-Unis et dans d'autres pays.

Cette distribution peut comprendre des composants développés par des tierces personnes.

Certaines composants de ce produit peuvent être dérivées du logiciel Berkeley BSD, licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays; elle est licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, le logo Solaris, le logo Java Coffee Cup, docs.sun.com, Java et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc., aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc. Ce produit comprend du logiciel développé par Computing Services à Carnegie Mellon University (<http://www.cmu.edu/computing>).

L'interface d'utilisation graphique OPEN LOOK et Sun a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui, en outre, se conforment aux licences écrites de Sun.

Les produits qui font l'objet de cette publication et les informations qu'il contient sont régis par la législation américaine en matière de contrôle des exportations et peuvent être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes chimiques ou biologiques ou pour le nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers des pays sous embargo des Etats-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exclusive, la liste de personnes qui font l'objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFACON.

Contents

Preface	37
1 Post-install Tasks and Layout	63
1.1 Creating UNIX System Users and Groups	63
▼ To Create UNIX System Users and Groups	64
1.2 To Prepare Directory Server for Messaging Server Configuration	64
1.3 Creating the Initial Messaging Server Runtime Configuration	65
1.3.1 Messaging Server Prerequisites	65
1.3.2 Messaging Server Configuration Checklist	65
▼ To Run the Configure Program	65
▼ To Perform a Silent Installation	70
1.4 Installing Messaging Server Against a Directory Server Replica	71
▼ To Install Messaging Server Against a Directory Server Replica	72
1.5 Installing Messaging Server Provisioning Tools	72
1.5.1 Schema 1 Delegated Administrator for Messaging	72
1.5.2 LDAP Provisioning Tools	74
1.6 SMTP Relay Blocking	74
1.7 Enabling Startup After a Reboot	76
▼ To Enable Messaging Server After a Reboot	76
1.8 Handling sendmail Clients	77
▼ To Obtain the Proper Version of the /usr/lib/sendmail on Solaris 8	77
▼ To Create the sendmail Configuration File on Solaris 9 Platforms	78
1.9 Configuring Messenger Express and Communications Express Mail Filters	79
1.10 Performance and Tuning	79
1.10.1 Java Message Queue (JMQ) Tuning	79
1.11 Post-Installation Directory Layout	80
1.12 Post-Installation Port Numbers	82
▼ To Change Port Numbers	83

2	Upgrading from Messaging Server 5.2 to Sun Java System Messaging Server	85
2.1	Information Moved	85
3	Configuring High Availability	87
3.1	Supported Versions	87
3.2	High Availability Models	87
3.2.1	Asymmetric	88
3.2.2	Symmetric	89
3.2.3	N+1 (N Over 1)	91
3.2.4	Choosing a High Availability Model	93
3.2.5	System Down Time Calculations	93
3.3	Installing Messaging Server High Availability—Overview	94
3.3.1	Cluster Agent Installation	94
3.3.2	Messaging Server and High Availability Notes	94
3.3.3	Using the useconfig Utility	95
3.4	Sun Cluster Installation	95
3.4.1	Sun Cluster Requirements	96
3.4.2	About HAStoragePlus	96
3.4.3	Configuring Messaging Server with Sun Cluster HAStorage or HAStoragePlus	97
3.4.4	Binding IP Addresses on a Server	114
3.4.5	Helpful Sun Cluster Commands to Manage Messaging HA	115
3.5	Veritas Cluster Server Agent Installation	116
3.5.1	Veritas Cluster Server Requirements	116
3.5.2	VCS Installation and Configuration Notes	116
3.5.3	MsgSrv Attributes and Arguments	119
3.6	Unconfiguring High Availability	119
	▼ To Unconfigure the Veritas Cluster Server	120
4	Configuring General Messaging Capabilities	121
4.1	To Modify Your Passwords	121
4.2	Managing Mail Users, Mailing Lists and Domains	122
	▼ To Remove a User from Messaging Server	123
	▼ To Remove a Domain from Messaging Server	123
4.3	Managing Messaging Server with Sun ONE Console	124
4.4	Starting and Stopping Services	124

4.4.1 To Start and Stop Services in an HA Environment	124
4.4.2 To Start and Stop Services in a non-HA Environment	125
4.4.3 Starting and Stopping a Messaging Server Running in MTA-only Mode	126
4.5 Automatic Restart of Failed or Unresponsive Services	127
4.5.1 Automatic Restart in High Availability Deployments	129
4.6 To Schedule Automatic Tasks	129
4.6.1 Scheduler Examples	130
4.6.2 Pre-defined Automatic Tasks	130
4.7 To Configure a Greeting Message	131
▼ To Create a New User Greeting	131
4.7.1 To Set a Per-Domain Greeting Message	131
4.8 To Set a User-Preferred Language	133
4.8.1 To Set a Domain Preferred Language	133
▼ To Specify a Site Language	134
4.9 To Customize Directory Lookups	134
▼ To Modify the Messaging Server LDAP User-lookup Settings	134
4.10 Encryption Settings	135
4.11 Setting a Failover LDAP Server	136
▼ To Set a Failover LDAP Server	136
4.12 Email Security Concerns	136
 5 Configuring POP, IMAP, and HTTP Services	137
5.1 General Configuration	137
5.1.1 Enabling and Disabling Services	138
5.1.2 Specifying Port Numbers	138
5.1.3 Ports for Encrypted Communications	139
5.1.4 Service Banner	139
5.2 Login Requirements	140
▼ To Set the Login Separator for POP Clients	140
5.2.1 To Allow Log In without Using the Domain Name	140
5.2.2 Password-Based Login	141
5.2.3 Certificate-Based Login	141
5.3 Performance Parameters	142
5.3.1 Number of Processes	142
5.3.2 Number of Connections per Process	143

5.3.3 Number of Threads per Process	144
5.3.4 Dropping Idle Connections	144
5.3.5 Logging Out HTTP Clients	145
5.4 Client Access Controls	145
5.5 To Configure POP Services	145
5.6 To Configure IMAP Services	146
5.6.1 Configuring IMAP IDLE	148
5.7 To Configure HTTP Services	151
5.7.1 Configuring Your HTTP Service	152
6 Enabling Single Sign-On (SSO)	157
6.1 Access Manager SSO for Sun Java System Servers	157
6.1.1 SSO Limitations and Notices	158
6.1.2 Configuring Messaging Server to Support SSO	158
6.1.3 Troubleshooting SSO	159
6.2 Trusted Circle SSO (Legacy)	160
6.2.1 Trusted Circle SSO Overview and Definitions	160
6.2.2 Trusted Circle SSO Applications	161
6.2.3 Trusted Circle SSO Limitations	161
6.2.4 Example Trusted Circle SSO Deployment Scenarios	161
6.2.5 Setting Up Trusted Circle SSO	163
6.2.6 Messenger Express Trusted SSO Configuration Parameters	167
7 Configuring and Administering Multiplexor Services	171
7.1 Multiplexor Services	171
7.1.1 Multiplexor Benefits	171
7.2 About Messaging Multiplexor	173
7.2.1 How the Messaging Multiplexor Works	173
7.2.2 Encryption (SSL) Option	175
7.2.3 Certificate-Based Client Authentication	175
7.2.4 User Pre-Authentication	176
7.2.5 MMP Virtual Domains	176
7.2.6 About SMTP Proxy	178
7.3 Setting Up the Messaging Multiplexor	178
7.3.1 Before You Configure MMP	179

7.3.2 Multiplexor Configuration	179
7.3.3 Multiplexor Files	180
7.3.4 Starting the Multiplexor	181
7.3.5 Modifying an Existing MMP	181
7.4 Configuring MMP with SSL	181
▼ To Configure MMP with SSL	181
▼ To Configure MMP with Client Certificate-based Login	182
7.4.1 A Sample Topology	183
7.5 MMP Tasks	186
7.5.1 To Configure Mail Access with MMP	187
7.5.2 To Set a Failover MMP LDAP Server	187
 8 MTA Concepts	189
8.1 The MTA Functionality	189
8.2 MTA Architecture and Message Flow Overview	193
8.2.1 Dispatcher and SMTP Server (Slave Program)	193
8.3 The Dispatcher	195
8.3.1 Creation and Expiration of Server Processes	195
8.3.2 To Start and Stop the Dispatcher	196
8.4 Rewrite Rules	196
8.5 Channels	197
8.5.1 Master and Slave Programs	197
8.5.2 Channel Message Queues	199
8.5.3 Channel Definitions	200
8.6 The MTA Directory Information	201
8.7 The Job Controller	202
8.7.1 To Start and Stop the Job Controller	204
 9 MTA Address Translation and Routing	205
9.1 The Direct LDAP Algorithm and Implementation	205
9.1.1 Domain Locality Determination	205
9.1.2 Alias expansion of local addresses	209
9.1.3 Processing the LDAP Result	214
9.1.4 To Modify Group Membership Attribute Syntax	228
9.2 Address Reversal	228

9.3 Asynchronous LDAP Operations	230
9.4 Settings Summary	231
9.5 Processing Multiple Different LDAP Attributes with the Same Semantics	232
10 About MTA Services and Configuration	233
10.1 Compiling the MTA Configuration	233
10.2 The MTA Configuration File	235
10.3 Mappings File	237
10.3.1 File Format in the Mappings File	239
10.3.2 Mapping Operations	241
10.4 Other MTA Configuration Files	251
10.4.1 Alias File	252
10.4.2 TCP/IP (SMTP) Channel Option Files	253
10.4.3 Conversion File	253
10.4.4 Dispatcher Configuration File	253
10.4.5 Mappings File	254
10.4.6 Option File	255
10.4.7 Tailor File	255
10.4.8 Job Controller File	256
10.5 Aliases	261
10.5.1 The Alias Database	262
10.5.2 The Alias File	262
10.5.3 Including Other Files in the Alias File	263
10.6 Command Line Utilities	263
10.7 SMTP Security and Access Control	263
10.8 Log Files	264
10.9 To Convert Addresses from an Internal Form to a Public Form	264
10.9.1 MTA Text Databases	265
10.9.2 To Set Address Reversal Controls	266
10.9.3 The Forward Lookup Table and FORWARD Address Mapping	268
10.10 Controlling Delivery Status Notification Messages	271
10.10.1 To Construct and Modify Status Notifications	272
10.10.2 To Customize and Localize Delivery Status Notification Messages	274
10.10.3 Internationalization of Generated Notices	276
10.10.4 Additional Status Notification Message Features	277

10.11 Controlling Message Disposition Notifications	284
10.11.1 To Customize and Localize Message Disposition Notification Messages	284
10.12 Optimizing MTA Performance	285
10.12.1 Optimizing Authorization Checks to the LDAP Directory for Messages Addressed to Mailing Lists	285
 11 Configuring Rewrite Rules	 289
11.1 Before You Begin	289
11.2 Rewrite Rule Structure	290
11.3 Rewrite Rule Patterns and Tags	291
11.3.1 A Rule to Match Percent Hacks	293
11.3.2 A Rule to Match Bang-Style (UUCP) Addresses	293
11.3.3 A Rule to Match Any Address	294
11.3.4 Tagged Rewrite Rule Sets	294
11.4 Rewrite Rule Templates	294
11.4.1 Ordinary Rewriting Templates, A%B@C or A@B	295
11.4.2 Repeated Rewrites Template, A%B	295
11.4.3 Specified Route Rewriting Templates, A@B@C@D or A@B@C	296
11.4.4 Case Sensitivity in Rewrite Rule Templates	296
11.5 How the MTA Applies Rewrite Rules to an Address	297
11.5.1 Step 1. Extract the First Host or Domain Specification	297
11.5.2 Step 2. Scan the Rewrite Rules	299
11.5.3 Step 3. Rewrite Address According to Template	300
11.5.4 Step 4. Finish the Rewrite Process	300
11.5.5 Rewrite Rule Failure	301
11.5.6 Syntax Checks After Rewrite	301
11.5.7 Handling Domain Literals	301
11.6 Template Substitutions and Rewrite Rule Control Sequences	302
11.6.1 Username and Subaddress Substitution, \$U, \$OU, \$IU	305
11.6.2 Host/Domain and IP Literal Substitutions, \$D, \$H, \$nD, \$nH, \$L	305
11.6.3 Literal Character Substitutions, \$\$, \$\$%, \$\$@	306
11.6.4 LDAP Query URL Substitutions, \$][.....	306
11.6.5 General Database Substitutions, \${...})	307
11.6.6 Apply Specified Mapping, \${...}	308
11.6.7 Customer-supplied Routine Substitutions, \${...}]	308

11.6.8 Single Field Substitutions, \$&, \$!, \$*, \$#	309
11.6.9 Unique String Substitutions	310
11.6.10 Source-Channel-Specific Rewrite Rules (\$M, \$N)	310
11.6.11 Destination-Channel-Specific Rewrite Rules (\$C, \$Q)	310
11.6.12 Direction-and-Location-Specific Rewrite Rules (\$B, \$E, \$F, \$R)	311
11.6.13 Host-Location-Specific Rewrites (\$A, \$P, \$S, \$X)	312
11.6.14 Changing the Current Tag Value, \$T	312
11.6.15 Controlling Error Messages Associated with Rewriting (\$?)	313
11.7 Handling Large Numbers of Rewrite Rules	314
11.8 Testing Rewrite Rules	314
11.9 Rewrite Rules Example	315
 12 Configuring Channel Definitions	 317
12.1 Configuring Channel Defaults	318
12.2 Channel Keywords Listed Alphabetically	318
12.3 Channel Keywords Categorized by Function	330
12.4 Configuring SMTP Channels	359
12.4.1 Configuring SMTP Channel Options	360
12.4.2 SMTP Command and Protocol Support	360
12.4.3 TCP/IP Connection and DNS Lookup Support	368
12.4.4 SMTP Authentication, SASL, and TLS	376
12.4.5 Using Authenticated Addresses from SMTP AUTH in Header	377
12.4.6 Support for SMTP Chunking	378
12.4.7 Specifying Microsoft Exchange Gateway Channels	379
12.4.8 Transport Layer Security	379
12.5 Configuring Message Processing and Delivery	380
12.5.1 Setting Channel Directionality	382
12.5.2 Implementing Deferred Delivery Dates	382
12.5.3 Specifying the Retry Frequency for Messages that Failed Delivery	383
12.5.4 Processing Pools for Channel Execution Jobs	384
12.5.5 Service Job Limits	384
12.5.6 Setting Connection Transaction Limits	386
12.5.7 Message Priority Based on Size	387
12.5.8 SMTP Channel Threads	387
12.5.9 Expansion of Multiple Addresses	388

12.5.10 Enable Service Conversions	389
12.6 Configuring Address Handling	389
12.6.1 Address Types and Conventions	389
12.6.2 Interpreting Addresses that Use ! and %	391
12.6.3 Adding Routing Information in Addresses	391
12.6.4 Disabling Rewriting of Explicit Routing Addresses	392
12.6.5 Address Rewriting Upon Message Dequeue	393
12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses	393
12.6.7 Legalizing Messages Without Recipient Header Lines	394
12.6.8 Stripping Illegal Blank Recipient Headers	395
12.6.9 Enabling Channel-Specific Use of the Reverse Database	395
12.6.10 Enabling Restricted Mailbox Encoding	395
12.6.11 Generating of Return-path Header Lines	396
12.6.12 Constructing Received Header Lines from Envelope To and From Addresses	396
12.6.13 Handling Comments in Address Header Lines	396
12.6.14 Handling Personal Names in Address Header Lines	397
12.6.15 Specifying Alias File and Alias Database Probes	398
12.6.16 Subaddress Handling	398
12.6.17 Enabling Channel-specific Rewrite Rules Checks	399
12.6.18 Removing Source Routes	399
12.6.19 Specifying Address Must be from an Alias	400
12.6.20 Recipient Address Handling	400
12.7 Configuring Header Handling	400
12.7.1 Rewriting Embedded Headers	401
12.7.2 Removing Selected Message Header Lines	401
12.7.3 Generating/Removing X-Envelope-to Header Lines	402
12.7.4 Converting Date to Two- or Four-Digits	402
12.7.5 Specifying Day of Week in Date	403
12.7.6 Automatic Splitting of Long Header Lines	403
12.7.7 Header Alignment and Folding	403
12.7.8 Specifying Maximum Length Header	404
12.7.9 Sensitivity Checking	404
12.7.10 Setting Default Language in Headers	405
12.7.11 Controlling <i>Message-hash:</i> Headers	405
12.8 Attachments and MIME Processing	405
12.8.1 Ignoring the Encoding Header Line	406

12.8.2 Automatic Defragmentation of Message/Partial Messages	406
12.8.3 Automatic Fragmentation of Large Messages	408
12.8.4 Imposing Message Line Length Restrictions	409
12.8.5 Interpreting Content-transfer-encoding Fields on Multiparts and Message/RFC822 Parts	410
12.9 Limits on Messages, Quotas, Recipients, and Authentication Attempts	410
12.9.1 Limits on Unsuccessful Authentication Attempts	410
12.9.2 Specifying Absolute Message Size Limits	411
12.9.3 Retargeting Messages Exceeding Limit on Size or Recipients	412
12.9.4 Handling Mail Delivery to Over Quota Users	414
12.9.5 Handling SMTP Mail with Lines Exceeding 1000 Characters	414
12.9.6 Controlling the Length of General and Filename Content-type and Content-disposition Parameters	414
12.9.7 Limiting Message Recipients	414
12.9.8 Limiting Header Size	415
12.10 File Creation in the MTA Queue	415
12.10.1 Controlling How Multiple Addresses on a Message are Handled	415
12.10.2 Spreading a Channel Message Queue Across Multiple Subdirectories	416
12.10.3 Setting Session Limits	416
12.11 Configuring Logging and Debugging	417
12.11.1 Logging Keywords	417
12.11.2 Debugging Keywords	417
12.11.3 Setting Loopcheck	418
12.12 Miscellaneous Keywords	418
12.12.1 Process Channel Overrides	418
12.12.2 Channel Operation Type	419
12.12.3 Pipe Channel	419
12.12.4 Specifying Mailbox Filter File Location	419
12.12.5 Spam Filter Keywords	420
12.12.6 Routing After Address Validation But Before Expansion	421
12.12.7 NO-SOLICIT SMTP Extension Support	424
12.12.8 Setting Limits on Bad RCPT TO Addresses	425
12.12.9 Set Channel Displays for Monitoring Framework	425
 13 Using Predefined Channels	 427
13.1 Predefined Channels	427

13.2 To Deliver Messages to Programs Using the Pipe Channel	429
13.3 To Configure the Native (/var/mail) Channel	430
13.4 To Temporarily Hold Messages Using the Hold Channel	431
13.5 The Conversion Channel	431
13.5.1 MIME Overview	432
13.5.2 Selecting Traffic for Conversion Processing	434
13.5.3 To Control Conversion Processing	435
13.5.4 To Bounce, Delete, Hold, Retry Messages Using the Conversion Channel Output	444
13.5.5 Conversion Channel Example	446
13.5.6 Automatic Arabic Character Set Detection	450
13.6 Character Set Conversion and Message Reformatting	452
13.6.1 Character Set Conversion	453
13.6.2 Message Reformatting	455
13.6.3 Service Conversions	460
 14 Integrating Spam and Virus Filtering Programs Into Messaging Server	463
14.1 Integrating Spam Filtering Programs Into Messaging Server—Theory of Operations ...	464
14.2 Deploying and Configuring Third Party Spam Filtering Programs	464
14.2.1 Loading and Configuring the Spam Filtering Software Client Library	465
14.2.2 Specifying the Messages to Be Filtered	466
14.2.3 Specifying Actions to Perform on Spam Messages	471
14.3 Using Symantec Brightmail Anti-Spam	476
14.3.1 How Brightmail Works	476
14.3.2 Brightmail Requirements and Performance Considerations	478
14.3.3 Deploying Brightmail	479
14.3.4 Brightmail Configuration Options	479
14.4 Using SpamAssassin	481
14.4.1 SpamAssassin Overview	481
14.4.2 SpamAssassin/Messaging Server Theory of Operations	481
14.4.3 SpamAssassin Requirements and Usage Considerations	482
14.4.4 Deploying SpamAssassin	483
14.4.5 SpamAssassin Configuration Examples	483
14.4.6 Testing SpamAssassin	489
14.4.7 SpamAssassin Options	491
14.5 Using Symantec Anti-Virus Scanning Engine (SAVSE)	493

14.5.1 SAVSE Overview	494
14.5.2 SAVSE Requirements and Usage Considerations	494
14.5.3 Deploying SAVSE	494
14.5.4 SAVSE Configuration Example	495
14.5.5 SAVSE Options	497
14.6 Using ClamAV	499
14.6.1 ClamAV/Messaging Server Theory of Operations	499
14.6.2 ClamAV Requirements and Usage Considerations	500
14.6.3 Deploying ClamAV	500
▼ To Jettison Virus– or Trojan Horse– Infected Email Using ClamAV	501
14.6.4 Testing ClamAV	502
14.6.5 ClamAV Options	503
14.7 Support for Sieve Extensions	504
14.8 Using Milter	506
14.8.1 Milter Overview	506
14.8.2 Milter/Messaging Server Theory of Operations	506
14.8.3 Milter Requirements and Usage Considerations	507
▼ To Deploy Milter	508
14.9 Cloudmark Anti-Abuse Client	509
14.10 Other Anti-Spam and Denial-of-Service Technologies	509
14.10.1 Anti-Spam Technique: Delay Sending the SMTP Banner	510
15 Handling Forged Email Using the Sender Policy Framework	511
15.1 Theory of Operations	511
15.2 Limitations	513
15.3 Pre-Deployment Considerations	514
15.4 Setting up the Technology	514
15.5 Reference Information	514
15.6 Testing SPF using spfquery	516
15.6.1 Syntax	516
15.6.2 Example with Debugging Enabled	517
15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)	518
16 LMTPT Delivery	521
16.1 LMTPT Delivery Features	522

16.2 Messaging Processing in a Two-Tier Deployment Without LMTP	522
16.3 Messaging Processing in a Two-Tier Deployment With LMTP	524
16.4 LMTP Overview	525
16.5 To Configure LMTP Delivery	525
▼ To Configure the Inbound MTA Relays with LMTP	526
16.5.1 To Configure Back End Stores with LMTP and a Minimal MTA	527
16.5.2 Configuring Relays for Sending Messages Via LMTP to Back End Systems with Message Stores and Full MTAs	529
16.5.3 Configuring LMTP on Back End Message Store Systems Having Full MTAs	529
16.5.4 Handling 4.2.1 Mailbox Busy Error in Response to LMTP Message Data	530
16.6 LMTP Protocol as Implemented	530
 17 Vacation Automatic Message Reply	 533
17.1 Vacation Autoreply Overview	533
17.2 Configuring Autoreply	534
17.2.1 Configuring Autoreply on the Back-end Store System	535
▼ To Configure Autoreply on a Relay	535
17.3 Vacation Autoreply Theory of Operation	536
17.4 Vacation Autoreply Attributes	537
17.5 Other Auto Reply Tasks and Issues	539
17.5.1 To Send Autoreply Messages for Email That Have Been Automatically Forwarded from Another Mail Server	539
 18 Mail Filtering and Access Control	 541
18.1 PART 1. MAPPING TABLES	541
18.2 Controlling Access with Mapping Tables	542
18.2.1 Access Control Mapping Tables—Operation	542
18.3 Access Control Mapping Table Flags	544
18.3.1 SEND_ACCESS and ORIG_SEND_ACCESS Tables	546
18.3.2 MAIL_ACCESS and ORIG_MAIL_ACCESS Mapping Tables	548
18.3.3 FROM_ACCESS Mapping Table	550
18.3.4 PORT_ACCESS Mapping Table	552
18.3.5 IP_ACCESS Mapping Table	554
18.3.6 To Limit Specified IP Address Connections to the MTA	555
18.4 When Access Controls Are Applied	556

18.5 To Test Access Control Mappings	556
18.6 To Add SMTP Relaying	557
18.6.1 Allowing SMTP Relaying for External Sites	559
18.7 Configuring SMTP Relay Blocking	560
18.7.1 How the MTA Differentiates Between Internal and External Mail	560
18.7.2 Differentiate Authenticated Users' Mail	562
18.7.3 Prevent Mail Relay	562
18.7.4 To Use DNS Lookups Including RBL Checking for SMTP Relay Blocking	563
18.8 Handling Large Numbers of Access Entries	566
18.9 PART 2. MAILBOX FILTERS	568
18.10 Sieve Filter Support	568
18.11 Sieve Filtering Overview	570
18.12 To Create User-level Filters	570
18.13 To Create Channel-level Filters	570
▼ To Create a Channel-level Filter	572
18.14 To Create MTA-Wide Filters	573
▼ To Create MTA-Wide Filters	573
18.14.1 Routing Discarded Messages Out the FILTER_DISCARD Channel	573
18.15 To Debug User-level Filters	574
▼ To Debug User-level Filters	574
18.15.1 imsimta test -exp Output	576
18.15.2 imsimta test -exp Syntax	577
 19 Throttling Incoming Connections Using MeterMaid	 579
19.1 Technical Overview	579
19.2 Theory of Operations	580
19.3 Configutil Parameters for MeterMaid	580
19.4 Limit Excessive IP Address Connections Using Metermiaid—Example	583
19.4.1 Additional Useful MeterMaid Options	585
 20 Managing the Message Store	 587
20.1 Overview	587
20.2 Message Store Directory Layout	589
20.2.1 Valid UIDs and Folder Names	592
20.3 How the Message Store Removes Messages	593

20.4 Specifying Administrator Access to the Store	594
▼ To Add an Administrator Entry	594
▼ To Modify an Administrator Entry	595
▼ To Delete an Administrator Entry	595
20.4.1 To Protect Mailboxes from Deletion or Renaming Except by an Administrator ...	595
20.5 About Shared Folders	596
20.6 Shared Folder Tasks	598
▼ To Specify Sharing Attributes for Private Shared Folders	598
▼ To Create a Public Shared Folder	599
20.6.1 To Add Shared Folders with an Email Group	600
20.6.2 To Set or Change a Shared Folder's Access Control Rights	601
20.6.3 To Enable or Disable Listing of Shared Folders	602
20.6.4 To Set Up Distributed Shared Folders	603
20.6.5 To Monitor and Maintain Shared Folder Data	605
20.7 Managing Message Types	606
20.7.1 Message Type Overview	607
▼ To Configure Message Types	608
20.7.2 Message Types in IMAP Commands	610
20.7.3 Sending Notification Messages for Message Types	612
20.7.4 Administering Quotas by Message Type	612
20.7.5 Expiring Messages by Message Type	614
20.8 About Message Store Quotas	616
20.8.1 Quota Overview	616
20.8.2 Quota Theory of Operations	617
20.8.3 Message Store Quota Attributes and Parameters	618
20.8.4 Configuring Message Store Quotas	620
20.9 To Set the Automatic Message Removal (Expire and Purge) Feature	625
20.9.1 imexpire Theory of Operation	625
20.9.2 To Deploy the Automatic Message Removal Feature	626
20.10 Configuring Message Store Partitions	635
20.10.1 To Add a Partition	636
20.10.2 To Move Mailboxes to a Different Disk Partition	636
20.10.3 Changing the Default Message Store Partition Definition	637
20.11 Performing Message Store Maintenance Procedures	638
20.11.1 Adding More Physical Disks to the Message Store	638
20.11.2 To Manage Mailboxes	638

20.11.3 Maximum Mailbox Size	641
20.11.4 To Monitor Quota Limits	641
20.11.5 To Monitor Disk Space	642
20.11.6 The <code>stored</code> Daemon	642
20.11.7 Reducing Message Store Size Due to Duplicate Storage of Identical Messages	643
20.12 Backing Up and Restoring the Message Store	646
20.12.1 Creating a Mailbox Backup Policy	647
20.12.2 To Create Backup Groups	648
20.12.3 Messaging Server Backup and Restore Utilities	649
20.12.4 Excluding Bulk Mail When You Perform Backups	650
20.12.5 Considerations for Partial Restore	651
20.12.6 To Use Legato Networker	653
20.12.7 To Use a Third Party Backup Software (Besides Legato)	656
20.12.8 Troubleshooting Backup and Restore Problems	657
20.12.9 Message Store Disaster Backup and Recovery	658
20.13 Monitoring User Access	659
20.14 Troubleshooting the Message Store	660
20.14.1 Standard Message Store Monitoring Procedures	661
20.14.2 Message Store Startup and Recovery	664
20.14.3 Repairing Mailboxes and the Mailboxes Database	667
20.14.4 Common Problems and Solutions	671
20.15 Migrating or Moving Mailboxes to a New System	675
20.15.1 Migrating User Mailboxes to Another Messaging Server While Online	676
▼ To Migrate User Mailboxes from One Messaging Server to Another While Online	678
▼ To Move Mailboxes Using an IMAP client	683
▼ To Move Mailboxes Using the <code>MoveUser</code> Command	684
▼ To Move Mailboxes Using the <code>imsimport</code> Command	685
21 Message Archiving	687
21.1 Archiving Overview	687
21.1.1 Message Archiving Systems: Compliance and Operational	688
22 Configuring the JMQ Notification Plug-in to Produce Messages for Message Queue	689
22.1 JMQ Notification Overview	689
22.1.1 Two Notification Messaging Services	689

22.1.2 Notification Plug-ins	690
22.1.3 Benefits of Using JMQ Notification	690
22.2 Configuring a JMQ Notification Service	692
22.2.1 Planning for Your JMQ Notification Service	692
▼ To Configure a JMQ Notification Plug-in	693
▼ To Configure Multiple Plug-ins	697
22.2.2 Specifying Notification Messages that Use More Than One <code>configutil</code> Parameter	698
22.3 JMQ Notification Messages and Properties	701
22.3.1 Notification Messages	701
22.3.2 Rules and Guidelines for Notification Messages	703
22.3.3 Notifications for Particular Message Types	703
22.3.4 Default Values of the <code>configutil</code> Parameters	704
22.3.5 Notification Message Properties	705
 23 Configuring Security and Access Control	713
23.1 About Server Security	713
23.2 About HTTP Security	714
23.3 Configuring Authentication Mechanisms	715
23.3.1 To Configure Access to Plaintext Passwords	717
23.3.2 To Transition Users	718
23.4 User Password Login	719
23.4.1 IMAP, POP, and HTTP Password Login	719
23.4.2 SMTP Password Login	720
23.5 Configuring Encryption and Certificate-Based Authentication	720
23.5.1 Obtaining Certificates	722
23.5.2 To Enable SSL and Selecting Ciphers	731
23.5.3 To Set Up Certificate-Based Login	733
23.5.4 How to Optimize SSL Performance Using the SMTP Proxy	734
23.6 Configuring Administrator Access to Messaging Server	735
23.6.1 Hierarchy of Delegated Administration	735
▼ To Provide Access to the Server as a Whole	736
23.6.2 To Restrict Access to Specific Tasks	736
23.7 Configuring Client Access to POP, IMAP, and HTTP Services	737
23.7.1 How Client Access Filters Work	738

23.7.2 Filter Syntax	739
23.7.3 Filter Examples	743
23.7.4 To Create Access Filters for Services	745
23.7.5 To Create Access Filters for HTTP Proxy Authentication	745
23.8 Enabling POP Before SMTP	746
▼ To Install the SMTP Proxy	746
23.9 Configuring Client Access to SMTP Services	749
23.10 User/Group Directory Lookups Over SSL	749
24 Administering S/MIME for Communications Express Mail	751
24.1 What is S/MIME?	751
24.1.1 Concepts You Need to Know	752
24.2 Required Software and Hardware Components	753
24.3 Requirements for Using S/MIME	754
24.3.1 Private and Public Keys	754
24.3.2 Keys Stored on Smart Cards	755
24.3.3 Keys Stored on the Client Machine	755
24.3.4 Publish Public Keys in LDAP Directory	755
24.3.5 Give Mail Users Permission to Use S/MIME	756
24.3.6 Multi-language Support	756
24.4 Getting Started After Installing Messaging Server	756
24.4.1 The S/MIME Applet	756
24.4.2 A Basic S/MIME Configuration	758
24.4.3 Accessing LDAP for Public Keys, CA certificates and CRLs Using Credentials	762
24.5 Parameters of the smime.conf File	764
24.6 Messaging Server Options	770
▼ To Set Messaging Server Options that Apply to S/MIME	770
24.7 Securing Internet Links With SSL	771
24.7.1 Securing the Link Between Messaging Server and Communications Express Mail	771
24.7.2 Securing the Link Between the Messaging Server and S/MIME Applet	772
24.8 Key Access Libraries for the Client Machines	772
24.8.1 Example	773
24.9 Verifying Private and Public Keys	774
24.9.1 Finding a User's Private or Public Key	775

24.9.2 When is a Certificate Checked Against a CRL?	776
24.9.3 Accessing a CRL	777
24.9.4 Proxy Server and CRL Checking	778
24.9.5 Using a Stale CRL	778
24.9.6 Determining Which Message Time to Use	779
24.9.7 Trouble Accessing a CRL	780
24.9.8 When a Certificate is Revoked	780
24.10 Granting Permission to Use S/MIME Features	781
24.10.1 S/MIME Permission Examples	781
24.11 Managing Certificates	782
24.11.1 CA Certificates in an LDAP Directory	782
24.11.2 Public Keys and Certificates in an LDAP Directory	783
24.11.3 Verifying That Keys and Certificates Exist in the LDAP Directory	784
24.11.4 Network Security Services Certificates	786
24.12 Communications Express S/MIME End User Information	786
24.12.1 Logging In for the First Time	787
24.12.2 Signature and Encryption Settings	788
24.12.3 Enabling the Java Console	789
 25 Managing Logging	 791
25.1 Overview of Logging	791
25.1.1 Types of Logging Data	792
25.1.2 Types of Messaging Server Log Files	792
25.1.3 Tracking a Message Across the Various Log Files	794
25.2 Tools for Managing Logging	795
25.3 Managing MTA Message and Connection Logs	796
25.3.1 Understanding the MTA Log Entry Format	796
25.3.2 Enabling MTA Logging	800
25.3.3 Specifying Additional MTA Logging Options	801
25.3.4 MTA Message Logging Examples	805
25.3.5 Enabling Dispatcher Debugging	818
25.4 Managing Message Store, Admin, and Default Service Logs	820
25.4.1 Understanding Service Log Characteristics	821
25.4.2 Understanding Service Log File Format	823
25.4.3 Defining and Setting Service Logging Options	824

25.4.4 Searching and Viewing Service Logs	826
25.4.5 Working With Service Logs	827
25.4.6 Using Message Tracing for Message Store Logging	830
25.4.7 Other Message Store Logging Features	831
25.4.8 Message Store Logging Examples	831
26 Troubleshooting the MTA	835
26.1 Troubleshooting Overview	835
26.2 Standard MTA Troubleshooting Procedures	836
26.2.1 Check the MTA Configuration	836
26.2.2 Check the Message Queue Directories	836
26.2.3 Check the Ownership of Critical Files	837
26.2.4 Check that the Job Controller and Dispatcher are Running	837
26.2.5 Check the Log Files	838
26.2.6 Run a Channel Program Manually	839
26.2.7 Starting and Stopping Individual Channels	840
26.2.8 An MTA Troubleshooting Example	842
26.3 Common MTA Problems and Solutions	846
26.3.1 TLS Problems	846
26.3.2 Changes to Configuration Files or MTA Databases Do Not Take Effect	847
26.3.3 The MTA Sends Outgoing Mail but Does Not Receive Incoming Mail	847
26.3.4 Dispatcher (SMTP Server) Won't Start Up	847
26.3.5 Timeouts on Incoming SMTP connections	848
26.3.6 Messages are Not Dequeued	849
26.3.7 MTA Messages are Not Delivered	852
26.3.8 Messages are Looping	853
26.3.9 Received Message is Encoded	856
26.3.10 Server-Side Rules (SSR) Are Not Working	857
26.3.11 Slow Response After Users Press Send Email Button	858
26.3.12 Asterisks in the Local Parts of Addresses or Received Fields	859
26.4 General Error Messages	859
26.4.1 Errors in mm_init	859
26.4.2 Compiled Configuration Version Mismatch	863
26.4.3 Swap Space Errors	863
26.4.4 File open or create errors	863

26.4.5 Illegal Host/Domain Errors	864
26.4.6 Errors in SMTP channels, os_smtp_* errors	865
27 Monitoring Messaging Server	867
27.1 Automatic Monitoring and Restart	867
27.2 Daily Monitoring Tasks	868
27.2.1 Checking postmaster Mail	868
27.2.2 Monitoring and Maintaining the Log Files	868
27.2.3 Setting Up the msprobe Utility	868
27.3 Monitoring System Performance	869
27.3.1 Monitoring End-to-end Message Delivery Times	869
27.3.2 Monitoring Disk Space	869
27.3.3 Monitoring CPU Usage	872
27.4 Monitoring the MTA	872
27.4.1 Monitoring the Size of the Message Queues	872
27.4.2 Monitoring Rate of Delivery Failure	873
27.4.3 Monitoring Inbound SMTP Connections	873
27.4.4 Monitoring the Dispatcher and Job Controller Processes	874
27.5 Monitoring LDAP Directory Server	875
27.5.1 Monitoring slapd	875
27.6 Monitoring Message Access	875
27.6.1 Monitoring imapd, popd and httpd	875
27.7 Monitoring the Message Store	877
27.7.1 Monitoring stored	877
27.7.2 Monitoring the State of Message Store Database Locks	878
27.8 Utilities and Tools for Monitoring	878
27.8.1 immonitor-access	878
27.8.2 imcheck	879
27.8.3 counterutil	879
27.8.4 Log Files	882
27.8.5 imsmta counters	882
27.8.6 imsmta qm counters	885
27.8.7 MTA Monitoring Using SNMP	885
27.8.8 imquotacheck for Mailbox Quota Checking	886
27.8.9 Monitoring Using msprobe and watcher Functions	886

A	SNMP Support	891
A.1	SNMP Implementation	891
A.1.1	SNMP Operation in the Messaging Server	892
A.2	Configuring SNMP Support for Messaging Server on Solaris 9	893
A.3	Configuring SNMP Support for Solaris 10 OS	894
A.3.1	Net-SNMP Configuration	895
A.3.2	Messaging Server Subagent Configuration	896
A.3.3	Running as a Standalone SNMP Agent	897
A.3.4	Monitoring Multiple Instances of Messaging Server	897
A.3.5	Using Standalone Agents for High-availability Failover	898
A.3.6	Distinguishing Multiple Instances Through SNMP v3 Context Names	898
A.3.7	Messaging Server's Net-SNMP-based SNMP Subagent Options	899
A.4	Monitoring from an SNMP Client	902
A.5	SNMP Information from the Messaging Server	903
A.5.1	applTable	903
A.5.2	assocTable	905
A.5.3	mtaTable	906
A.5.4	mtaGroupTable	907
A.5.5	mtaGroupAssociationTable	908
A.5.6	mtaGroupErrorTable	909
B	Administering Event Notification Service in Messaging Server	911
B.1	Loading the ENS Publisher in Messaging Server	911
▼	To Load the ENS Publisher on Messaging Server	912
B.2	Running Sample Event Notification Service Programs	912
▼	To Run the Sample ENS Programs	912
B.3	Administering Event Notification Service	913
B.3.1	Starting and Stopping ENS	913
B.3.2	Event Notification Service Configuration Parameters	913
C	Short Message Service (SMS)	915
C.1	Introduction	915
C.1.1	One-Way SMS	916
C.1.2	Requirements	917
C.2	SMS Channel Theory of Operation	918

C.2.1 Directing Email to the Channel	918
C.2.2 The Email to SMS Conversion Process	919
C.2.3 The SMS Message Submission Process	925
C.2.4 Site-defined Address Validity Checks and Translations	929
C.2.5 Site-defined Text Conversions	930
C.3 SMS Channel Configuration	934
C.3.1 Adding an SMS Channel	935
C.3.2 Creating an SMS Channel Option File	937
C.3.3 Available Options	938
C.3.4 Adding Additional SMS Channels	958
C.3.5 Adjusting the Frequency of Delivery Retries	959
C.3.6 Sample One-Way Configuration (MobileWay)	960
C.3.7 Configuring the SMS Channel for Two-Way SMS	962
C.4 SMS Gateway Server Theory of Operation	963
C.4.1 Function of the SMS Gateway Server	963
C.4.2 Behavior of the SMPP Relay and Server	963
C.4.3 Remote SMPP to Gateway SMPP Communication	964
C.4.4 SMS Reply and Notification Handling	965
C.5 SMS Gateway Server Configuration	967
C.5.1 Setting Up Bidirectional SMS Routing	967
C.5.2 Enabling and Disabling the SMS Gateway Server	968
C.5.3 Starting and Stopping the SMS Gateway Server	968
C.5.4 SMS Gateway Server Configuration File	969
C.5.5 Configuring Email-To-Mobile on the Gateway Server	969
C.5.6 Configuring Mobile-to-Email Operation	971
C.5.7 Configuration Options	973
C.5.8 Global Options	973
C.5.9 SMPP Relay Options	977
C.5.10 SMPP Server Options	980
C.5.11 Gateway Profile Options	981
C.5.12 Configuration Example for Two-Way SMS	986
C.6 SMS Gateway Server Storage Requirements	989
 D Installation Worksheets	991
D.1 Directory Server Installation	991

D.2 Directory Server Setup Script (comm_dssetup.pl)	993
D.3 Messaging Server Initial Runtime Configuration	994
Glossary	997
Index	999

Figures

FIGURE 3–1	Asymmetric High Availability Mode	88
FIGURE 3–2	Symmetric High Availability Mode	90
FIGURE 3–3	N + 1 High Availability Mode	92
FIGURE 3–4	A Simple Messaging Server HA Configuration	98
FIGURE 3–5	Veritas Cluster Server Dependency Tree 1	117
FIGURE 3–6	Veritas Cluster Dependency Tree	118
FIGURE 5–1	HTTP Service Components	151
FIGURE 6–1	Simple SSO Deployment	162
FIGURE 6–2	Complex SSO Deployment	163
FIGURE 7–1	Clients and Servers in an MMP Installation	174
FIGURE 7–2	Multiple MMPs Supporting Multiple Messaging Servers	184
FIGURE 8–1	Messaging Server, Simplified Components View (Communications Express not Shown)	191
FIGURE 8–2	MTA Architecture	192
FIGURE 8–3	Master and Slave Programs	199
FIGURE 8–4	ims-ms Channel	199
FIGURE 14–1	Brightmail and Messaging Server Architecture	477
FIGURE 16–1	Two Tier Deployment Without LMTP	523
FIGURE 16–2	Two Tier Deployment With LMTP	524
FIGURE 20–1	Message Store Directory Layout	590
FIGURE 20–2	Example of Shared Mail Folder List as Seen from a Mail Client	597
FIGURE 20–3	Distributed Shared Folders—Example	604
FIGURE 20–4	Message Store Digest Repository	644
FIGURE 23–1	Encrypted Communications with Messaging Server	721
FIGURE 24–1	S/MIME Applet	757
FIGURE 24–2	Verifying Private and Public Keys.	775
FIGURE A–1	SNMP Information Flow	893
FIGURE C–1	Logical Flow For One-Way and Two-Way SMS	916
FIGURE C–2	SMS Channel Email Processing	921

FIGURE C-3	SMS Channel Email Processing (<i>continued</i>)	923
------------	---	-----

Tables

TABLE 1-1	Post-Installation Directories and Files	80
TABLE 1-2	Port Numbers Designated During Installation	82
TABLE 1-3	Potential Port Number Conflicts	82
TABLE 3-1	HA Model Comparison	93
TABLE 3-2	HA Down Probabilities	93
TABLE 3-3	Veritas Cluster Server Attributes	119
TABLE 3-4	MsgSrv Arguments	119
TABLE 4-1	Passwords Set in Messaging Server Initial Runtime Configuration	122
TABLE 4-2	Start, Stop, Restart in a Sun Cluster 3.0/3.1 Environment	124
TABLE 4-3	Start, Stop, Restart in Veritas 3.5, 4.0, 4.1 and 5.0 Environments	124
TABLE 4-4	Services Monitored by <i>watcher</i> and <i>msprobe</i>	127
TABLE 4-5	HA Automatic Restart Parameters	129
TABLE 6-1	Access Manager Single Sign-On Parameters	158
TABLE 6-2	SSO Interoperability	161
TABLE 6-3	Trusted Circle Single Sign-On Parameters	168
TABLE 7-1	Messaging Multiplexor Configuration Files	180
TABLE 7-2	MMP Commands	181
TABLE 9-1	Object Classes Resulting from Various <i>schematag</i> Values	215
TABLE 9-2	Attributes to Check Against	216
TABLE 9-3	MTA Options Which Set the Retrieved Disk Quota and Message Quota Attributes	219
TABLE 9-4	MTA Options, Default Attributes, and Metacharacters	219
TABLE 9-5	Single-Character Prefixes for options in the DELIVERY_OPTIONS MTA option	221
TABLE 9-6	Additional Metacharacters for Use in Delivery Options	221
TABLE 9-7	Integers Controlling Behavior Modification of the \$nI and \$nS Metacharacters	222
TABLE 9-8	Special Template Strings	223
TABLE 9-9	Group Expansion Default Attributes and MTA Option to Set	225

TABLE 9-10	local.imta.schematag Values and Attributes	229
TABLE 9-11	Settings for the LDAP_USE_ASYNC MTA Option	231
TABLE 10-1	Addresses and Associated Channels	237
TABLE 10-2	Messaging Server Mapping Tables	238
TABLE 10-3	Mapping Pattern Wildcards	241
TABLE 10-4	Mapping Template Substitutions and Metacharacters	244
TABLE 10-5	MTA Configuration Files	252
TABLE 10-6	Job Controller Configuration File Options	260
TABLE 10-7	REVERSE mapping table flags	265
TABLE 10-8	FORWARD Output Mapping Table Flags Description	269
TABLE 10-9	FORWARD Input Mapping Table Flags Description	269
TABLE 10-10	Notification Message Substitution Sequences	273
TABLE 10-11	Delivery Status and Message Disposition Notification Options	277
TABLE 10-12	Keywords for Sending Notification Messages to the Postmaster and Sender ..	282
TABLE 11-1	Summary of Special Patterns for Rewrite Rules	293
TABLE 11-2	Summary of Template Formats for Rewrite Rules	294
TABLE 11-3	Extracted Addresses and Host Names	298
TABLE 11-4	Summary of Rewrite Rule Template Substitutions and Control Sequences	303
TABLE 11-5	LDAP URL Substitution Sequences	307
TABLE 11-6	Single Field Substitutions	309
TABLE 11-7	Sample Addresses and Rewrites	315
TABLE 12-1	Alphabetized List of Channel Keywords	318
TABLE 12-2	Address Handling Keywords	331
TABLE 12-3	Attachments and MIME Processing	334
TABLE 12-4	Character Sets and Eighth Bit Data	335
TABLE 12-5	File Creation in the MTA Queue Area	335
TABLE 12-6	Header Keywords	336
TABLE 12-7	Incoming Channel Matching and Switching Keywords	341
TABLE 12-8	Logging and Debugging Channel Keywords	341
TABLE 12-9	Long Address Lists or Headers Channel Keywords	342
TABLE 12-10	Mailbox Filter Channel Keywords	342
TABLE 12-11	NO-SOLICIT SMTP Extension Support Keywords	344
TABLE 12-12	Notification and Postmaster Messages Keywords	344
TABLE 12-13	Processing Control and Job Submission Keywords	346
TABLE 12-14	Sensitivity Limit Keywords	347
TABLE 12-15	Keywords for Limits on Messages, User Quotas, Privileges, and Authentication	

TABLE 12–16	Attempts	348
TABLE 12–17	SMTP Authentication, SASL and TLS Keywords	350
TABLE 12–18	SMTP Commands and Protocol Keywords	352
TABLE 12–19	TCP/IP Connection and DNS Lookup Support Keywords	355
TABLE 12–20	Miscellaneous Keywords	357
TABLE 12–21	SMTP Channels	359
TABLE 12–22	SMTP Command and Protocol Keywords	361
TABLE 12–23	TCP/IP Connection and DNS Lookup Keywords	369
TABLE 12–24	authrewrite Bit Values	377
TABLE 12–25	Message Processing and Delivery Keywords	381
TABLE 13–1	missingrecipientpolicy Values	394
TABLE 13–2	Predefined Channels	427
TABLE 13–3	Local Channel Options	430
TABLE 13–4	Conversion Channel Environment Variables	439
TABLE 13–5	Conversion Channel Output Options	442
TABLE 13–6	Special Directives Commonly Used By the Conversion Channel	445
TABLE 13–7	Conversion Parameters	447
TABLE 14–1	CHARSET-CONVERSION Mapping Table Keywords	453
TABLE 14–2	MTA Spam Filter Options (option.dat)	473
TABLE 14–3	Selected Brightmail Configuration File Options	479
TABLE 14–4	SpamAssassin Options (spamassassin.opt)	491
TABLE 14–5	Returned String for the SpamAssassin mode Option	493
TABLE 14–6	ICAP Options	497
TABLE 14–7	Returned Verdict String for the ICAP mode Option	499
TABLE 15–1	ClamAV Options	503
TABLE 15–2	SPF Processing Results	512
TABLE 15–3	SPF Keywords	514
TABLE 15–4	SPF Limiting Options	515
TABLE 15–5	SPF Failure and Error Options	516
TABLE 16–1	spfquery options	517
TABLE 17–1	LMTP Status Codes for Recipients	531
TABLE 18–1	Prefix Characters for the Autoreply Rule in DELIVERY_OPTIONS	534
TABLE 18–2	Access Control Mapping Tables	543
TABLE 18–3	Access Mapping Flags and Metacharacters	544
TABLE 18–4	PORT_ACCESS Mapping Flags	553
TABLE 18–4	IP_ACCESS Mapping Table Flags	554

TABLE 18-5	filter Channel Keyword URL-pattern Substitution Tags (Case-insensitive)	570
TABLE 20-1	Message Store Command-line Utilities	588
TABLE 20-2	Message Store Directory Description	591
TABLE 20-3	ACL Rights Characters	601
TABLE 20-4	Variables for Configuring Distributed Shared Folders	603
TABLE 20-5	readership Options	605
TABLE 20-6	Message Store Quota Attributes	618
TABLE 20-7	Message Store configutil parameters	619
TABLE 20-8	imexpire Attributes	629
TABLE 20-9	imexpire Folder Patterns Using Regular Expressions	633
TABLE 20-10	Expire and Purge configutil Log and Scheduling Parameters	634
TABLE 20-11	relinker configutil Parameters	646
TABLE 20-12	stored Operations	663
TABLE 20-13	Message Store Database Snapshot Parameters	666
TABLE 20-14	reconstruct Options	667
TABLE 22-1	JMQ Notification Messages	702
TABLE 22-2	configutil Parameters and Their Default Values	705
TABLE 22-3	Standard Notification Message Properties	706
TABLE 22-4	Properties Specific to Particular Notification Messages	706
TABLE 22-5	Properties Carried with Each Notification Message	711
TABLE 23-1	Some SASL and SASL-related configutil Parameters	716
TABLE 23-2	SSL Ciphers for Messaging Server	732
TABLE 23-3	Wildcard Names for Service Filters	741
TABLE 24-1	Required Hardware and Software for Client Machine	753
TABLE 24-2	Required Software for Server Machine	753
TABLE 24-3	S/MIME Configuration Parameters in smime.conf File	764
TABLE 24-4	Special Libraries for the Client Machines	773
TABLE 24-5	Signature and Encryption Checkboxes of Communications Express Mail	788
TABLE 25-1	Messaging Server Log Files	793
TABLE 25-2	Logging Entry Action Codes	797
TABLE 25-3	Logging Entry Modifier Codes	798
TABLE 25-4	SMTP Channel's LOG_CONNECTION Action Codes + or - Entries	799
TABLE 25-5	Dispatcher Debugging Bits	819
TABLE 25-6	Logging Levels for Store and Administration Services	821
TABLE 25-7	Categories in Which Log Events Occur	822

TABLE 25–8	Store and Administration Log File Components	823
TABLE 26–1	MTA Log Files	839
TABLE 27–1	counterutil alarm Statistics	880
TABLE 27–2	counterutil imapstat Statistics	881
TABLE 27–3	counterutil diskstat Statistics	881
TABLE 27–4	counterutil serverresponse Statistics	882
TABLE 27–5	msprobe and watcher configutil Options	887
TABLE 27–6	Useful Alarm Message configutil Parameters	889
TABLE A–1	SNMP Subagent Options	899
TABLE B–1	iBiff Configuration Parameters	913
TABLE C–1	SMS Attributes	918
TABLE C–2	Fields in Generated in a BIND_TRANSMITTER PDU	926
TABLE C–3	Mandatory Fields in Generated SUBMIT_SM PDUs	927
TABLE C–4	Optional Fields in Generated SUBMIT_SM PDUs	928
TABLE C–5	SMS Channel Options	938
TABLE C–6	USE_HEADER_FROM Values	944
TABLE C–7	Valid Values for USE_UCS2	945
TABLE C–8	Numeric Plan Indicator Values	945
TABLE C–9	Typical TON Values	946
TABLE C–10	SMS Priority Values Interpreted for Each SMS Profile Type	947
TABLE C–11	Mapping for Translating Priority Header to SMS Priority Flags	948
TABLE C–12	Result of Values for DEFAULT_PRIVACY and USE_HEADER_SENSITIVITY	948
TABLE C–13	SMS Interpretation of Privacy Values	949
TABLE C–14	Mapping Translation of Sensitivity Headers to SMS Privacy Values	949
TABLE C–15	DEFAULT_VALIDITY_PERIOD Format and Values	950
TABLE C–16	DEBUG Bitmask	956
TABLE C–17	Substitution Sequences	957
TABLE C–18	Two-Way Configuration Exceptions	962
TABLE C–19	SMPP Server Protocol Data Units	964
TABLE C–20	Global Options	973
TABLE C–21	DEBUG Bitmask	976
TABLE C–22	SMPP Relay Options	978
TABLE C–23	SMPP Server Options	980
TABLE C–24	SMS Gateway Server Profile Options	982
TABLE C–25	Priority Flag Mapping from SMS to Email	985
TABLE C–26	Privacy Flags Mapping from SMS to Email	986

TABLE C-27	SMS Gateway Server Storage Requirements	989
TABLE D-1	Directory Server Installation Parameters	991
TABLE D-2	comm_dssetup.pl Script Parameters	993
TABLE D-3	Initial Runtime Configuration Parameters	995

Examples

EXAMPLE 1-1	Changing the Messenger Express HTTP Port Number	83
EXAMPLE 10-1	Sample Job Controller Configuration File in UNIX	257
EXAMPLE 13-1	conversions File Entry	435
EXAMPLE 13-2	Converting ISO-8859-1 to UTF-8 and back	454
EXAMPLE 13-3	Converting EUC-JP to ISO-2022-JP and Back	455
EXAMPLE 14-1	Example LDAP User Entry for Brightmail	467
EXAMPLE 14-2	Example LDAP Domain Entry for Brightmail	469
EXAMPLE 18-1	SEND_ACCESS Mapping Table	547
EXAMPLE 18-2	MAIL_ACCESS Mapping Table	549
EXAMPLE 18-3	FROM_ACCESS Mapping Table	550
EXAMPLE 18-4	imsimta test -exp Output	576
EXAMPLE 20-1	IMAP FETCH Session Based on the Message-Type configutil Configurations	611
EXAMPLE 20-2	IMAP SEARCH Session Based on the Message-Type configutil Configurations	611
EXAMPLE 20-3	Sample Rules for Expiring Different Message Types	615
EXAMPLE 20-4	Example imexpire Rules	632
EXAMPLE 25-1	MTA Logging: A Local User Sends An Outgoing Message	806
EXAMPLE 25-2	MTA Logging – Including Optional Logging Fields	807
EXAMPLE 25-3	MTA Logging – Sending to a List	808
EXAMPLE 25-4	MTA Logging – Sending to a Nonexistent Domain	809
EXAMPLE 25-5	MTA Logging – Sending to a Nonexistent Remote User	811
EXAMPLE 25-6	MTA Logging – Rejecting a Remote Side's Attempt to Submit a Message	812
EXAMPLE 25-7	MTA Logging – Multiple Delivery Attempts	813
EXAMPLE 25-8	MTA Logging – Incoming SMTP Message Routed Through the Conversion Channel	814
EXAMPLE 25-9	MTA Logging: Outbound Connection Logging	815
EXAMPLE 25-10	MTA Logging – Inbound Connection Logging	818
EXAMPLE 25-11	Message Store Logging – Invalid Password	832

EXAMPLE 25-12	Message Store Logging – Account Disabled	832
EXAMPLE 25-13	Message Store Logging – Append	832
EXAMPLE 25-14	Message Store Logging – Message Retrieved by a Client	832
EXAMPLE 25-15	Message Store Logging Example: Message Removed from a Folder	832
EXAMPLE 25-16	Message Store Logging – Login	833
EXAMPLE C-1	Example SMS_TEXT Mapping Table.	931
EXAMPLE C-2	Language Specification Part of Channel Option File	953

Preface

This guide explains how to administer the Sun Java™ System Messaging Server and its accompanying software components. Messaging Server provides a powerful and flexible cross-platform solution to meet the email needs of enterprises and messaging hosts of all sizes using open Internet standards.

For revision history of this document, see “[Sun Java System Messaging Server 6.3 Administration Guide Revision History](#)” on page 44.

Who Should Use This Book

You should read this book if you are responsible for administering and deploying Messaging Server at your site. You should also have read the [*Sun Java Communications Suite 5 Deployment Planning Guide*](#).

Before You Read This Book

This book assumes that you are responsible for administering the Messaging Server software and that you have a general understanding of the following:

- The Internet and the World Wide Web
- Messaging Server protocols
- Sun Java System Directory Server and LDAP
- System administration and networking
- General deployment architectures

How This Book Is Organized

This manual contains the following chapters and appendix:

TABLE P-1 How This Book Is Organized

Chapter	Description
Preface	General information about using this book.
Chapter 1, “Post-install Tasks and Layout”	Describes the tasks required to get you to a point where you have a functioning Messaging Server.
Chapter 2, “Upgrading from Messaging Server 5.2 to Sun Java System Messaging Server”	Describes how to upgrade from Messaging Server 5.2 to this version of Messaging Server.
Chapter 3, “Configuring High Availability”	Provides information on how to configure the Veritas Cluster Server and Sun Cluster high availability clustering software for use with the Messaging Server.
Chapter 4, “Configuring General Messaging Capabilities”	Describes the general Messaging Server tasks.
Chapter 5, “Configuring POP, IMAP, and HTTP Services”	Describes how to configure your server to support POP, IMAP and HTTP services
Chapter 6, “Enabling Single Sign-On (SSO)”	Explains how to enable Single Sign-On.
Chapter 7, “Configuring and Administering Multiplexor Services”	Describes the Messaging Multiplexor (MMP) for standard mail protocols (POP, IMAP and SMTP).
Chapter 8, “MTA Concepts”	Provides a conceptual description of the MTA.
Chapter 9, “MTA Address Translation and Routing”	Describes Address Translation and Routing.
Chapter 10, “About MTA Services and Configuration”	Describes MTA services and configuration.
Chapter 11, “Configuring Rewrite Rules”	Describes how to configure rewrite rules in the imta.cnf file.
Chapter 12, “Configuring Channel Definitions”	Explains how to use channel keyword definitions in the MTA configuration file imta.cnf.
Chapter 13, “Using Predefined Channels”	Describes how to use pre-defined channel definitions in the MTA.
Chapter 14, “Integrating Spam and Virus Filtering Programs Into Messaging Server”	Describes how to integrate and configure spam and virus filtering software with Messaging Server.
Chapter 15, “Handling Forged Email Using the Sender Policy Framework”	Describes a technology that can detect and reject forged email during the SMTP dialogue.

TABLE P-1 How This Book Is Organized *(Continued)*

Chapter	Description
Chapter 16, “LMTP Delivery”	Describes LMTP operation and deployment.
Chapter 17, “Vacation Automatic Message Reply”	Describes the vacation autoreply mechanism.
Chapter 18, “Mail Filtering and Access Control”	Discusses how to filter mail based on its source (sender, IP address and so on) or header strings.
Chapter 19, “Throttling Incoming Connections Using MeterMaid”	Describes a repository process that supplants <code>conn_throttle.so</code> , providing similar functionality but extending it across the Messaging Server installation.
Chapter 20, “Managing the Message Store”	Describes the message store and its administration interface.
Chapter 21, “Message Archiving”	Describes archiving concepts for Messaging Server.
Chapter 22, “Configuring the JMQ Notification Plug-in to Produce Messages for Message Queue”	Describes how to configure a JMQ notification plug-in to produce messages to be consumed by clients in a Message Queue service.
Chapter 23, “Configuring Security and Access Control”	Describes how to configure security and access control to the messaging server.
Chapter 24, “Administering S/MIME for Communications Express Mail”	Describes how to administer S/MIME.
Chapter 25, “Managing Logging”	Describes Messaging Server logging facility.
Chapter 26, “Troubleshooting the MTA”	Describes common tools, methods, and procedures for troubleshooting the MTA.
Chapter 27, “Monitoring Messaging Server”	Describes the monitoring of the Messaging Server.
Appendix A, “SNMP Support”	Describes how to enable SNMP support for the Messaging Server.
Appendix B, “Administering Event Notification Service in Messaging Server”	Describes how to enable the Event Notification Service Publisher (ENS Publisher) and administer Event Notification Service (ENS) in Messaging Server.
Appendix C, “Short Message Service (SMS)”	Describes how to implement the Short Message Service (SMS).
Appendix D, “Installation Worksheets”	Provides worksheets by which you can plan your installation.

Messaging Server Documentation Set

The following table summarizes the books included in the Messaging Server core documentation set.

TABLE P-2 Messaging Server Documentation

Document Title	Contents
<i>Sun Java System Messaging Server 6.3 Administration Reference</i>	Provides detailed reference information on Messaging Server commands, configutil parameters, configuration files and options, and supported standards.
<i>Sun Java Communications Suite 5 Deployment Planning Guide</i>	Contains the information you need to deploy Sun Java System Communications Services including Messaging Server.
<i>Sun Java System Delegated Administrator 6.4 Administration Guide</i>	Explains how to configure and administer Sun Java System Communications Services Delegated Administrator. Also describes the Delegated Administrator commands.
<i>Sun Java Communications Suite 5 Schema Migration Guide</i>	Describes how to migrate Sun Java System LDAP Directory data from LDAP Schema 1 to LDAP Schema 2 for System Messaging Server and Calendar Server.
<i>Sun Java Communications Suite 5 Event Notification Service Guide</i>	Describes the Event Notification Service (ENS) architecture and APIs for Messaging Server and Calendar Server. It gives detailed instructions on the ENS APIs that you can use to customize your server installation.
<i>Sun Java Communications Suite 5 Release Notes</i>	Contains important information available at the time of release of Sun Java System Messaging Server. New features and enhancements, known issues and limitations, and other information are also addressed here.
<i>Sun Java Communications Suite 5 Schema Reference</i>	Serves as a reference for schema information for Messaging Server and Calendar Server.
<i>Sun Java System Communications Express 6.3 Administration Guide</i>	Describes how to administer Communications Express and its accompanying software components.
<i>Sun Java System Communications Express 6.3 Customization Guide</i>	Explains how to customize the look and feel of Communications Express. Focuses on how to perform the most commonly requested customizations.
<i>Sun Java Enterprise System 5 Installation Guide for UNIX</i>	Contains the information you need to install the Sun Java Enterprise System (Java ES) software.

TABLE P-2 Messaging Server Documentation *(Continued)*

Document Title	Contents
<i>Sun Java System Messaging Server 6 2005Q4 MTA Developer's Reference</i>	Describes the Messaging Server Message Transfer Agent (MTA) Software Development Kit (SDK) and Callable Send facility.
<i>Sun Java Enterprise System Glossary</i>	Glossary.
<i>Sun Java Communications Suite 5 Documentation Center</i>	Topical links to Communications Suite Documentation.

In addition, use the following URL to see the documentation that applies to all Communications Services products: (<http://www.sun.com/bigadmin/hubs/comms/>)

Related Books

The <http://docs.sun.com> web site enables you to access Sun technical documentation online. You can browse the archive or search for a specific book title or subject.

For other server documentation related to deploying Messaging Server, go to the following:

- Access Manager documentation: <http://docs.sun.com/app/docs/coll/1292.2>
- Calendar Server documentation: <http://docs.sun.com/app/docs/coll/1313.2>
- Communications Express documentation: <http://docs.sun.com/app/docs/coll/1312.2>
- Directory Server documentation: <http://docs.sun.com/app/docs/coll/1224.3>
- Instant Messaging documentation: <http://docs.sun.com/app/docs/coll/1309.2>
- Messaging Server documentation: <http://docs.sun.com/app/docs/coll/1312.2>

Default Path and File Names

The following table describes the default path and file name that are used in this book.

TABLE P-3 Default Paths and File Names

Placeholder	Description	Default Value
<i>msg-svr-base</i>	Represents the base installation directory for Messaging Server. The Messaging Server default base installation and product directory depends on your specific platform.	Solaris systems: /opt/SUNWmsgsr Linux systems: /opt/sun/messaging

Typographic Conventions

The following table describes the typographic changes that are used in this book.

TABLE P-4 Typographic Conventions

Typeface	Meaning	Example
AaBbCc123	The names of commands, files, and directories, and onscreen computer output	Edit your .login file. Use ls -a to list all files. machine_name% you have mail.
AaBbCc123	What you type, contrasted with onscreen computer output	machine_name% su Password:
AaBbCc123	A placeholder to be replaced with a real name or value	The command to remove a file is rm <i>filename</i> .
AaBbCc123	Book titles, new terms, and terms to be emphasized (note that some emphasized items appear bold online)	Read Chapter 6 in the <i>User's Guide</i> . A <i>cache</i> is a copy that is stored locally. Do <i>not</i> save the file.

Shell Prompts in Command Examples

The following table shows default system prompts and superuser prompts.

TABLE P-5 Shell Prompts

Shell	Prompt
C shell on UNIX and Linux systems	machine_name%
C shell superuser on UNIX and Linux systems	machine_name#
Bourne shell and Korn shell on UNIX and Linux systems	\$
Bourne shell and Korn shell superuser on UNIX and Linux systems	#

Symbol Conventions

The following table explains symbols that might be used in this book.

TABLE P–6 Symbol Conventions

Symbol	Description	Example	Meaning
[]	Contains optional arguments and command options.	ls [-l]	The -l option is not required.
{ }	Contains a set of choices for a required command option.	-d {y n}	The -d option requires that you use either the y argument or the n argument.
\${ }	Indicates a variable reference.	\${com.sun.javaRoot}	References the value of the com.sun.javaRoot variable.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.
→	Indicates menu item selection in a graphical user interface.	File → New → Templates	From the File menu, choose New. From the New submenu, choose Templates.

Accessing Sun Resources Online

The docs.sun.com web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. Books are available as online files in PDF and HTML formats. Both formats are readable by assistive technologies for users with disabilities.

To access the following Sun resources, go to <http://www.sun.com>.

- Downloads of Sun products
- Services and solutions
- Support (including patches and updates)
- Training
- Research
- Communities (for example, Sun Developer Network)

Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to <http://docs.sun.com> and click Send Comments. In the online form, provide the full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL.

Sun Java System Messaging Server 6.3 Administration Guide Revision History

Version	Date	Description of Changes
14	15 February 2008	Fixed a number of documentation bugs. Changes are listed below.
13	22 July 2007	<ul style="list-style-type: none">■ Added a list of sections changed from version 10 to Version 13 (see “Administration Guide Revision Changes from Version 10 to 12” on page 51). Most changes were made in version 12, but this list makes it easier to note what changed between version releases.
12	8 June 2007	<ul style="list-style-type: none">■ Incorporated and/or clarified “New MTA Features” in <i>Sun Java Communications Suite 5 Release Notes</i>.■ Added bug fixes and other small documentation fixes.

Version	Date	Description of Changes
11	14 April 2007	<ul style="list-style-type: none"> ■ Added a Chapter 19, “Throttling Incoming Connections Using MeterMaid,” example. ■ Updated Veritas Server Cluster supported versions to 3.5, 4.0, 4.1 and 5.0. ■ Documented new “12.12.5 Spam Filter Keywords” on page 420. ■ Added bug fixes and other small documentation fixes.
10	March 2007	Initial release of this technical note.

Administration Guide Revision Changes from Version 14 to 15

These are Administration Guide changes from 819-4428-14 to 819-4428-15. <> angle bracketed words indicate deleted text with replaced by words immediately following. [] bracketed words indicated new text.

[Chapter 1, “Post-install Tasks and Layout”](#)

[“To Run the Configure Program” on page 65](#)

In this section, added the following note regarding Solaris 10 at the end of the discussion of adding the FQDN in the `/etc/hosts/` and related files:

Note – On Solaris OS 10 U3 and earlier platforms, you not only have to add the Fully Qualified Domain Name (FQDN) to the `/etc/hosts` file, but also to the `/etc/inet/ipnodes` file. Otherwise, you will get an error indicating that your host name is not a Fully Qualified Domain Name. From Solaris OS 10U4 onwards, the contents of the `/etc/inet/ipnodes` and `/etc/hosts` files have been merged together into just the `/etc/hosts` file. Applying kernel patch 120011-14 on any Solaris 10 system will also perform the merge, and subsequent removal of the `/etc/inet/ipnodes` file.

New level 2 section: [“1.10.1 Java Message Queue \(JMQ\) Tuning” on page 79.](#)

[Chapter 11, “Configuring Rewrite Rules”](#)

Added the following paragraph to 11.4.2. Repeated Rewrites Template, A%B: [Note, however, that while the special A%B form does cause rewriting of the current domain to restart, it is actually just a continuation of the current rewriting process. It does not rewrite the entire process from the beginning. It does not perform the \$* pattern when it goes through the second time.]

[Chapter 12, “Configuring Channel Definitions”](#)

Removed `immnougent`

[“12.4.1 Configuring SMTP Channel Options” on page 360](#)

Added the following paragraph: <SMTP channel options that pertain to a destination channel should be placed in the appropriate channel option file (that is, `tcp_local_option`, `tcp_auth_option`, `tcp_intranet_option`). SMTP channel option files that pertain to a source channel MUST be placed in the `tcp_local_option` file since all messages come into the MTA via the `tcp_local` channel before any channel switching takes place for incoming messages.>

[Chapter 14, “Integrating Spam and Virus Filtering Programs Into Messaging Server”](#)

Removed references to the `SOCKS_USERNAME` and `SOCKS_PASSWORD` options from the book. The references appeared in the following sections in Chapter 14:

[“14.4.7 SpamAssassin Options” on page 491](#)

[“14.5.5 SAVSE Options” on page 497](#)

[“14.6.5 ClamAV Options” on page 503](#)

[“To Deploy Milter” on page 508](#)

[Chapter 16, “LMTP Delivery”](#)

Removed all references and instructions regarding the LMTP native channel.

[Chapter 18, “Mail Filtering and Access Control”](#)

[“Processing Control \(\\$C, \\$L, \\$R, \\$E\)” on page 246](#)

Rewrote this section.

[Table 18–3](#)

Changed description of `$T text` to the following: If bit 1 (value 2) of the `LOG_CONNECTION` MTA option is set and the `$N` flag is set so that the connection is rejected, then `$T` outputs the entire right hand side text in a “T” record. The T log entry will include the entire mapping result string (`$N` and its string). In contrast, bit 4 of `LOG_CONNECTION` is a different effect: it will cause material after two vertical bars to be included in normal “C” (connection close) records.

[“18.3.4 PORT_ACCESS Mapping Table” on page 552](#)

Added a new table. Fixed other table.

[“18.3.5 IP_ACCESS Mapping Table” on page 554](#)

`<source-channel|address-count|address-current|ip-current|hostname>source-channel|address-`

[“18.4 When Access Controls Are Applied” on page 556](#)

Added new information at the bottom of this section about when access control mapping tables are applied.

[“18.7.4 To Use DNS Lookups Including RBL Checking for SMTP Relay Blocking” on page 563](#)

Fixed some errors in the code and added new text added regarding the PORT_ACCESS table being probed both by the dispatcher, when accepting connections, and by the `tcp_smtp_server` process under certain circumstances.

[Chapter 19, “Throttling Incoming Connections Using MeterMaid”](#)

New note added to the bottom of [“19.3 Configutil Parameters for MeterMaid” on page 580](#)

[Chapter 20, “Managing the Message Store”](#)

Globally replaced `moveuser` with `MoveUser`. Note capitalization.

[“To Add an Email Group to a Shared Folder” on page 600](#)

```
readership -s user/gregk/<tennis>gardening tennis@sesta.com lrp
```

[Table 20–9](#)

<Age of message in days before being expunged. (integer)>Number of days in the message store before being expunged. (integer)

[“20.14.4.9 IMAP Events Become Slow” on page 675](#)

Brand new section.

[“Expiration Rules Guidelines” on page 628](#)

Added new material to the Note: This is still true, but expire rules using header constraints (example: expiring a message with a specific subject line) are not supported. <Also, regular expressions in the expire rules created with configutil need to be POSIX compliant rules. If you want to use UNIX compliant regular expressions you will have to use the `store.expire` file. In addition, using both configutil options and the global `store.expirerule` configuration file is not supported. If the configuration file is present, configutil options are not used.> In any case, it is best to use `store.expirerule` to specify all expiration rules.

[“20.11.4 To Monitor Quota Limits” on page 641](#)

Removed `imquotacheck -i`. No longer valid.

[“20.10.2 To Move Mailboxes to a Different Disk Partition” on page 636](#)

<Make sure user is disconnected from their mailbox during the migration process . . .> The user does not have to be disconnected from their mailbox during this migration process.

[“20.12.8 Troubleshooting Backup and Restore Problems” on page 657](#) and [“To Migrate User Mailboxes from One Messaging Server to Another While Online” on page 678](#)

Added the following text: When `ims restore` or any processing intensive operation takes significantly more system resources than normal, and continues doing so longer than the `msprobe` interval, there may be a temporary backlog of DB transaction log files to be cleared. If there are more files than specified in `local.store.maxlog`, then `msprobe` may erroneously restart all the processes during a restore. To prevent this from happening, disable `msprobe` during the `ims restore`.

[“20.14.4.1 Reduced Message Store Performance” on page 671](#)

Added this section, which basically cross-references [Tuning the mboxlist Database Cache \(`http://wikis.sun.com/display/CommSuite/Tuning+the+mboxlist+Database+Cache`\)](#)

[“To Move Mailboxes Using an IMAP client” on page 683](#)

In step 2, changed `<local.store.relinker to enable>` to [Set `local.store.relinker.enabled`] to yes]

[“To Move Mailboxes Using the `MoveUser` Command” on page 684](#)

In step 2, changed `<local.store.relinker to enable>` to [Set `local.store.relinker.enabled`] to yes]

[“To Move Mailboxes Using the `imsimport` Command” on page 685](#)

In step 2, changed `<local.store.relinker to enable>` to [Set `local.store.relinker.enabled`] to yes]

[Chapter 23, “Configuring Security and Access Control”](#)

[“23.7 Configuring Client Access to POP, IMAP, and HTTP Services” on page 737](#)

Discussed the LDAP method of configuring client access. Added cross-references to the LDAP attributes “`mailAllowedServiceAccess`” in [Sun Java Communications Suite 5 Schema Reference](#) and “`mailDomainAllowedServiceAccess`” in [Sun Java Communications Suite 5 Schema Reference](#)

[Chapter 24, “Administering S/MIME for Communications Express Mail”](#)

[“24.1 What is S/MIME?” on page 751](#)

Rewrote this section.

Administration Guide Revision Changes from Version 12 to 14

These are Administration Guide changes from 819-4428-12 to 819-4428-14. `<>` angle bracketed words indicate deleted text with replaced by words immediately following. [] bracketed words indicated new text. Note that only one change was between 12 and 13, and that was the addition of this Revision Guide History.

Chapter 3, “Configuring High Availability”

“4.5.1 Automatic Restart in High Availability Deployments” on page 129

Chapter 4, “Configuring General Messaging Capabilities”

Table 4–5, entry for local.autorestart.timeout <fails more than twice> fails more than once

“4.12 Email Security Concerns” on page 136

Added this new section which talks about disabling XADR, XCIR, XGEN, and XSTA.

Chapter 9, “MTA Address Translation and Routing”

“9.1.3.8 Delivery Options Processing” on page 220

In the example: <=\$M%\$\\> [=M%\$\\]

Chapter 10, “About MTA Services and Configuration”

“8.7 The Job Controller” on page 202

Added new information about setting MAX_MESSAGES.

Chapter 12, “Configuring Channel Definitions”

Table 12–21.s

smtp (This keyword is equivalent to <smtp_crCrLf> smtp_crlf.)

“12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362

<The keyword smtp_crlf means that lines must be terminated with a carriage return (CR) line feed (LF) sequence. The keyword smtp_lf or smtp means that an LF without a preceding CR is accepted. Finally, smtp_cr means that a CR is accepted without a following LF. These option affect only the handling of incoming material.>

[The keyword smtp_crlf or smtp means that lines must be terminated with a carriage return (CR) line feed (LF) sequence. The keyword smtp_lf means that an LF without a preceding CR is accepted as well as the standard CRLF sequence. The keyword smtp_cr means that a CR is accepted without a following LF. Finally, smtp_crCrLf means that any of CR, LF, or the standard CRLF sequence are allowed as the SMTP line terminator. These option affect only the handling of incoming material.]

“12.6.14 Handling Personal Names in Address Header Lines” on page 397

Added: If the PERSONAL_NAMES mapping table returns 8-bit characters, they are UTF-8 encoded.

Chapter 13, “Using Predefined Channels”

[“13.5 The Conversion Channel” on page 431](#)

Rewrote a number of small sections to make clearer. This in response to bug 4902284.

[Chapter 14, “Integrating Spam and Virus Filtering Programs Into Messaging Server”](#)[“14.2.3 Specifying Actions to Perform on Spam Messages” on page 471](#)

Table 14–1, MTA Spam Filter Options. Rewrote `spamfilterX_final`

[“14.9 Cloudmark Anti-Abuse Client” on page 509](#)

Mentioned support for Cloudmark anti-spam solution.

[“14.7 Support for Sieve Extensions” on page 504](#)

Added that `spamadjust` and `spamtest` can be also be used with Brightmail.

[Chapter 16, “LMTP Delivery”](#)[“16.5.1 To Configure Back End Stores with LMTP and a Minimal MTA” on page 527](#)

`tcp_lmtpss lmtp [flagtransfer]`

[Chapter 18, “Mail Filtering and Access Control”](#)[“18.3 Access Control Mapping Table Flags” on page 544](#)

Table 18–2 — Enhanced description of \$X.

[Chapter 20, “Managing the Message Store”](#)

Wherever `hashdir` was recommended as the command to use for finding the directory containing a specified folder, `mboxutil -lxp pattern` was recommended instead.

[“20.2.1 Valid UIDs and Folder Names” on page 592](#)

Rewrote this section and renamed it to *Valid UIDs and Folder Names*.

[Chapter 26, “Troubleshooting the MTA”](#)[“26.2.7 Starting and Stopping Individual Channels” on page 840](#)

Added the following:

NOTE - The command `imsimta qm start/stop channel` may fail if run simultaneously for many channels at the same time. The tool might have trouble updating the `hold_list` and could report: `QM-E-NOTSTOPPED`, `unable to stop the channel; cannot update the hold list.` `imsimta qm start/stop channel` should only be used sequentially with a few seconds interval between each run.

If you only want the channel to run between certain hours, use the following options in the channel definition section in the job controller configuration file:

```
urgent_delivery=08:00-20:00  
normal_delivery=08:00-20:00  
nonurgent_delivery=08:00-20:00
```

[Appendix C, “Short Message Service \(SMS\)”](#)

[“C.2.5.2 Message Body Entries” on page 931](#)

< These entries establish mappings to be applied to . . . > Text removed. Body mappings are not supported.

Administration Guide Revision Changes from Version 10 to 12

These are Administration Guide changes from 819-4428-10 to 819-4428-12. < > angle bracketed words indicate deleted text with replaced by words immediately following. [] bracketed words indicated new text.

Chapter 1, Post-install Tasks and Layout

[“1.6 SMTP Relay Blocking” on page 74](#)

- 1) <\$(192.45.67.89/24)> \$(192.45.67.89/32)
- 2) <first 24> full 32

[“1.8 Handling sendmail Clients” on page 77](#)

When you <upgraded>installed previous versions of Messaging Server, the /usr/lib/sendmail binary was replaced with a component of the <sendmail>Messaging Server product. In Messaging Server, Messaging Server {6.0 to the current version,} this replacement during <upgrade>install is no longer <occurs>necessary. Therefore, you may need to obtain the proper version of the /usr/lib/sendmail binary from the most current sendmail patch.

```
<FEATURE("nulclient", "smtp:rhino.west.sesta.com')dnl  
MASQUERADE_AS("west.sesta.com')dnl  
define("confDOMAIN_NAME", "west.sesta.com')dnl>
```

Replaced by:

```
FEATURE('nullclient', 'smtp:rhino.west.sesta.com')dnl  
MASQUERADE_AS('west.sesta.com')dnl  
define('confDOMAIN_NAME', 'west.sesta.com')dnl
```

Chapter 2, Upgrading from Messaging Server 5.2 to Sun Java System Messaging Server

All of this moved to the technical article entitled: Upgrading from Messaging Server 5.2 to Sun Java System Messaging Server (<http://www.sun.com/bigadmin/hubs/comms/files/ms52-ms6-upgrade.html>).

Chapter 3, Configuring High Availability

“3.1 Supported Versions” on page 87 Moved to Release Notes

“3.4.1 Sun Cluster Requirements” on page 96. Version requirements are moved to Release Notes.

Chapter 4, Configuring General Messaging Capabilities

“4.4.1 To Start and Stop Services in an HA Environment” on page 124

To Start, Stop, Restart in Veritas <1.3, 2.0, 2.1,>3.5, 4.0, 4.1 and <3.5> 5.0 Environments

“4.4.2 To Start and Stop Services in a non-HA Environment” on page 125. Minor typos.

“4.4.3 Starting and Stopping a Messaging Server Running in MTA-only Mode” on page 126. New section!

“4.6 To Schedule Automatic Tasks” on page 129

1) <A fully qualified command pathname is required.>Paths can be relative to msg-svr-base or absolute paths. See Pre-defined Automatic Tasks for relative path examples.

2) <send SIGHUP to> refresh the scheduler process: <kill -HUP> refresh sched [scheduler_pid]

“4.6.1 Scheduler Examples” on page 130

<20,40,60> 0,20,40

“4.8 To Set a User-Preferred Language” on page 133

<accept-language> Accept-Language (twice!)

Chapter 5, Configuring POP, IMAP, and HTTP Services

“5.2.1 To Allow Log In without Using the Domain Name” on page 140

<inetdomainsearchfilter> inetDomainSearchFilter

“5.2.3 Certificate-Based Login” on page 141

<You don't need to disable password login to enable certificate-based login. If password login is enabled, and if> If

“5.6.1.1 Prerequisites” on page 148

<Ibiff> iBiff (twice)

[“To Configure IMAP IDLE” on page 148](#)

<As mailsrv run> Run:

```
<local.store.notifyplugin.ensHost> local.store.notifyplugin.enshost
    <local.store.notifyplugin.ensHost> local.store.notifyplugin.enshost
    <local.store.notifyplugin.ensEventKey> local.store.notifyplugin.enseventkey
    <local.store.notifyplugin.ensEventKey> local.store.notifyplugin.enseventkey
```

[Chapter 6, Enabling Single Sign-On \(SSO\)](#)

No changes.

[Chapter 7, Configuring and Administering Multiplexor Services](#)

[“7.2.3 Certificate-Based Client Authentication” on page 175](#)

<certmap>certmap.conf

[“To Configure MMP with SSL” on page 181 \(Version 13\)](#)

Steps 2, 3, Note removed:

<2. If you have installed the Admin Server ...>

<3. Since the sslpassword.conf file is set ...>

< NOTE: An alternative approach to steps 1-8 ...>

Steps 5 & 6 consolidated into a single step: [If you do not want to use SSL between the ...]

[“7.4.1.1 IMAP Configuration Example” on page 184](#)

1) </opt/SUNWmsgsr/config/cert7.db> /opt/SUNWmsgsr/config/cert8.db

2) <"".> /opt/SUNWmsgsr/config/sslpassword.conf

[“7.4.1.2 POP Configuration Example” on page 185](#)

<It also provides a spoof message file.>

[“7.5.1 To Configure Mail Access with MMP” on page 187](#)

<is not configured automatically, it has to be explicitly configured. In addition, the MMP>

[“7.5.2 To Set a Failover MMP LDAP Server” on page 187](#)

1) <IMAPProxyAservice.cfg>ImapProxyAservice.cfg

2) </o=INTERNET"> /o=internet

3) [Make sure there is a space between the host names in the above configuration.]

Chapter 8, MTA Concepts

No substantive changes.

Chapter 9, MTA Address Translation and Routing

[“9.1.1.2 Domain Map Determination of Domain Locality” on page 207](#)

Appended to end of section:

[Two MTA options support more efficient domain lookups from user base domain names. They are LDAP_BASEDN_FILTER_SCHEMA1, which is a string specifying a filter used to identify Schema 1 domains when performing user base domain name searches. The default is the value of LDAP_DOMAIN_FILTER_SCHEMA1 if that MTA option is specified. If neither option is specified the default is (objectclass=inetDomain). LDAP_BASEDN_FILTER_SCHEMA2 is a string specifying additional filter elements used to identify Schema 2 domains when performing user base domain name searches. The default is the value of LDAP_DOMAIN_FILTER_SCHEMA2, if that MTA option is specified. If neither option is specified, the default is an empty string.]

[“9.1.2.2 The \\$V Metacharacter” on page 210](#)

Added to bulleted MTA option list: LDAP_DOMAIN_ATTR_CATCHALL_MAPPING (no default value)

[“9.1.3.12 Optin and Presence Attributes” on page 223](#)

1) Rewrote the first sentence to: The LDAP_OPTIN1 through LDAP_OPTIN8 MTA options specify LDAP attributes for per-user spam filter opt-in values based on destination addresses.

2) Added the following to first paragraph: LDAP_SOURCE_OPTIN1 through LDAP_SOURCE_OPTIN8 provide comparable originator-address-based per-user spam filter optins.

[“9.1.4 To Modify Group Membership Attribute Syntax” on page 228](#) New section.

Chapter 10, About MTA Services and Configuration

Table 10–1

<c_channel>b_channel <d_channel>a_channel

[“10.3.1 File Format in the Mappings File” on page 239](#)

1) In entire chapter: <use_text_database> use_text_databases

2) <252> 256 and 1024 characters respectively. [The maximum size of a line in the mapping file is 4096.]

Table 10–4

In entry for \$E: [+\$1E exits immediately without interpreting the rest of the template.]

Table 10–6

In entry MAX_MESSAGES [The minimum value is 10.]

[“10.9 To Convert Addresses from an Internal Form to a Public Form” on page 264](#)

<NOTE: Messaging Server provides other facilities for address manipulation, such as the aliases file and specialized mapping tables. For best performance, however, rewrite rules should be used whenever possible to perform address manipulations. See Chapter 11, Configuring Rewrite Rules.>

[Messaging Server provides other facilities for address manipulation, such as the aliases file and specialized mapping tables. For best performance, however, rewrite rules should be used whenever possible to perform address manipulations. See Chapter 11, Configuring Rewrite Rules.]

Table 10–8

Added. Completely new.

[“10.10.4.1 To Block Content Return on Large Messages” on page 278](#)

Appended at end:

[The MTA fetches the block limit associated with the envelope return address and will set RET=HDRS if no return policy is specified and the message size exceeds the block limit. This prevents nondelivery reports for large messages from being undeliverable themselves. No new options or settings are associated with this change.]

[“10.12 Optimizing MTA Performance” on page 285.](#) New section.

[“10.12.1 Optimizing Authorization Checks to the LDAP Directory for Messages Addressed to Mailing Lists” on page 285.](#) New section.

Chapter 11, Configuring Rewrite Rules**Table 11–4**

Added entry for \$nT: Overrides the default ALIAS_MAGIC setting, where n is an appropriate value for the ALIAS_MAGIC MTA option. Overrides the setting for the domain when the rule matches during alias expansion.

Table 11–5

Added entry for \$. See manual.

Chapter 12, Configuring Channel Definitions

New Channels: addressssrs, caption, chunkingclient, chunkingserver, description, destinationsspamfilterX, destinationsrs, disabledestinationsspamfilterX, disablesourcesspamfilterX, ignoremessageencoding, ignoremultipartencoding, interpretmessageencoding, interpretmultipartencoding, noaddressssrs, nochunkingclient, nochunkingserver, nodestinationsrs, nosourcesrs, sourcesspamfilterX, sourcesrs.

New sections: “12.4.6 Support for SMTP Chunking” on page 378, “12.6.20 Recipient Address Handling” on page 400, “12.8.5 Interpreting Content-transfer-encoding Fields on Multiparts and Message/RFC822 Parts” on page 410, “12.12.9 Set Channel Displays for Monitoring Framework” on page 425

“12.12.5 Spam Filter Keywords” on page 420. Rewrote and added new keywords.

Chapter 13, Using Predefined Channels

Table 13–1

tcp_local entry adds the following: [Sometimes tcp_local gets mail from remote SMTP hosts via proxy or firewall. tcp_local is also sometimes used for internal relay activities.]

“Mail Conversion Tags” on page 440

Added a lot of new material including a new section “Including Conversion Tag Information in Various Mapping Probes” on page 441.

Chapter 14, Integrating Spam and Virus Filtering Programs Into Messaging Server

Table 14–1

LDAP_optinX rewritten. LDAP_SOURCE_OPTINX added.

Table 14–3

New entry for USERNAME_MAPPING.

“To Specify User-level Filtering” on page 466

New Note added to Step 1.

“To Specify Domain-level Filtering” on page 468. New Note added to Step 1.

“14.7 Support for Sieve Extensions” on page 504. New paragraph added (3rd from top).

“To Deploy Milter” on page 508

```
<spamfilterX_config_file=/opt/SUNWmsgsr/lib/milter.opt>
spamfilter1_config_file=/opt/SUNWmsgsr/lib/milter.opt
```

“14.10 Other Anti-Spam and Denial-of-Service Technologies” on page 509. New section.

“14.10.1 Anti-Spam Technique: Delay Sending the SMTP Banner” on page 510. New Section

Chapter 15, Handling Forged Email Using the Sender Policy Framework

“15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518. New Section

Chapter 16, LMTP Delivery

“16.5.1 To Configure Back End Stores with LMTP and a Minimal MTA” on page 527. Rewritten.

“16.5.4 Handling 4.2.1 Mailbox Busy Error in Response to LMTP Message Data” on page 530. New section

Chapter 17, Vacation Automatic Message Reply

“17.4 Vacation Autoreply Attributes” on page 537

New attribute definition: LDAP_AUTOREPLY_ADDRESSES

“17.5.1 To Send Autoreply Messages for Email That Have Been Automatically Forwarded from Another Mail Server” on page 539. New section.

Chapter 18, Mail Filtering and Access Control

Table 18–2

New entry for \$! (Available in FROM_ACCESS only). Disables the sending of vacation messages regarding this message; that is, it sets the novacation flag.

“18.3.3 FROM_ACCESS Mapping Table” on page 550

Added paragraph to end of section: [The \$(metacharacter in a FROM_ACCESS specifies that an address should be read from the result string and used to replace the current overriding postmaster address. \$) has the same effect with the added constraint that the overriding postmaster address must not be set prior to invoking the mapping. This allows for specific postmaster addresses to be used with addresses in nonlocal domains - domain postmaster addresses by definition only work with locally defined domains. The override address is (currently) the last string read from the FROM_ACCESS result prior to reading any \$N/\$F failure result.]

Table 18–3

New Entry for \$D: Causes an additional argument to be read from the template result after the mandatory SMTP auth rulset and realm, and optional application information addition. This value must be an integer with the same semantics as the BANNER_PURGE_DELAY value. That is, it specifies the number of centiseconds to delay before purging and sending the banner. A value of

0 disabled both the delay and purge. Note that any PORT_ACCESS mapping setting overrides the BANNER_PURGE_DELAY SMTP channel option. See Anti-Spam Technique: Delay Sending the SMTP Banner for details on using this anti-spam feature. \$U [Selectively] enable channel level debugging.

[“18.3.6 To Limit Specified IP Address Connections to the MTA” on page 555](#)

First paragraph rewritten: To limit how often a particular IP address can connect to the MTA, see [Chapter 19, Throttling Incoming Connections Using MeterMaid](#). Limiting connections by particular IP addresses can be useful for preventing excessive connections used in denial-of-service attacks. In the past, this function was performed using the shared library, conn_throttle.so in the Port Access mapping table. No new enhancements are planned for conn_throttle.so and MeterMaid is its more effective replacement.

[“18.10 Sieve Filter Support” on page 568](#)

Two new bullets:

- Sieve redirect can now add three header fields:

```
resent-date: date-of-resend-operation  
resent-to: address-specified-in-redirect  
resent-from: address-of-sieve-owner
```

The new :resent and :noresent arguments to redirect can be used to control whether or not these fields are added. If neither argument is specific the system default is used. The system default is controlled by the new SIEVE_REDIRECT_ADD_RESENT MTAoption. Setting the option to 1 causes these fields to be generated unless :noresent used. A setting of 0 causes the fields to be generated only if :resent is used. The option defaults to 1, which means the fields are generated by default for regular redirects.

- Sieve redirect has been enhanced with three new arguments:

:resetmailfrom - Reset the envelope FROM: address to that of the current Sieve owner.

:keepmailfrom - Preserve the envelope FROM: address from the original message.

:notify - Specify a new set of notification flags for the redirected message. A single parameter is required giving a list of notification flags. The same set of flags accepted by the NOTIFY parameter of the DSN SMTP extension are accepted here: SUCCESS, FAILURE, DELAY and NEVER. Note that these flags are specified as a Sieve list, for example:

```
redirect :notify ["SUCCESS","FAILURE"] "foo@example.com";
```

The default if :notify isn't specified as the normal SMTP default of FAILURE, DELAY. :keepmailfrom is the default unless :notify is specified, in which case the default switches to :resetmailfrom. The one additional exception is that specification of the SUCCESS flag forces the use of :resetmailfrom unconditionally.

Chapter 19, Throttling Incoming Connections Using MeterMaid

Rewritten for clarity. Also added two new sections:

“19.4 Limit Excessive IP Address Connections Using Metermaid—Example” on page 583 and
“19.4.1 Additional Useful MeterMaid Options” on page 585

Chapter 20, Managing the Message Store

Changed three instances of `store.overquotastatus` to `local.store.overquotastatus` in Table 20–7 and “20.8.4.4 To Enable or Disable Quota Enforcement” on page 623.

“20.14.1.4 Check stored Processes” on page 662

`<stored -t -v> imcheck`

“20.15 Migrating or Moving Mailboxes to a New System” on page 675

This entire section was moved from Chapter 2, Upgrading from Messaging Server 5.2 to Sun Java System Messaging Server.

Chapter 21, Message Archiving No changes.

Chapter 22, Configuring the JMQ Notification Plug-in to Produce Messages for Message Queue

“To Configure a JMQ Notification Plug-in” on page 693.

`<o Sun Java Messaging Server 6 2006Q3>`

Chapter 23, Configuring Security and Access Control

“23.1 About Server Security” on page 713

For “End-user account configuration” bullet, `<product (valid only for Sun LDAP Schema 1).>`

Chapter 24, Administering S/MIME for Communications Express Mail

No substantive changes

Chapter 25, Managing Logging

“25.3.1 Understanding the MTA Log Entry Format” on page 796

Step 5: [The SMS channel can be configured to log a page count rather than file size in this field.
See `LOG_PAGE_COUNT`.]

Table 25–3

`<P POP-before-SMTP via the MMP was used. P is added to the E record.>`

[“25.3.3 Specifying Additional MTA Logging Options” on page 801](#)

[“To Send MTA Logs to syslog” on page 801](#)

<A value of 0 is the default and indicates that syslog (event log) logging is not performed.> [A value of 0 disables generation of the syslog notices. A non-zero value enables generation of the syslog notices, with the absolute value controlling the syslog priority and facility mask. (Positive values mean syslog notices and the regular mail.log* entries; negative values, which are not recommended, mean syslog notices only, disabling the regular mail.log* entries. A value of 0 is the default and indicates that syslog (event log) logging is not performed.]

[“To Control Formatting of Log Entries” on page 802](#)

1) Add to enqueue/dequeue attributes:

[qt - the amount of time a message has spent in the queue (LOG_QUEUE_TIME=1)]

2) Add to connection attributes: [ct - the amount of time a message has spent in the queue (LOG_QUEUE_TIME=1, also used in en entries)]

3) Updated the following MTA Logging examples:

[“25.3.4.5 MTA Logging Example – Sending to a Nonexistent Remote User” on page 810](#)

[“25.3.4.6 MTA Logging Example – Rejecting a Remote Side’s Attempt to Submit a Message” on page 812](#)

[“25.3.4.7 MTA Logging Example – Multiple Delivery Attempts” on page 813](#)

[“25.3.4.8 MTA Logging – Incoming SMTP Message Routed Through the Conversion Channel” on page 814](#) — Outbound Connection Logging <process ID here is the same, 40a5> [process ID here is the same, 1f625] [10. The connection mailhub.sesta.com is closed now that the delivery of the message (dave in this example) is complete.]

[“25.3.4.10 MTA Logging Example: Inbound Connection Logging” on page 817](#)

[“To Enable Message Tracing” on page 830](#)

1) [configutil -o local.msgtrace.active -v "msgtrace" In this command message trace information for all processes is written to the msgtrace log file]

2) Remove the next two sections: <To Redirect Message Tracing to a Single Log File> and <To Unconfigure Message Trace Loggings>

Chapter 26, Troubleshooting the MTA

[“26.2.4 Check that the Job Controller and Dispatcher are Running” on page 837](#)
26.2.4 Check that the Job Controller and Dispatcher are Running

You could also use `imsimta qm` jobs to list, channel by channel, all active and pending delivery processing jobs currently being managed by the Job Controller. Additional cumulative information is provided for each channel such as the number of message files successfully delivered and those requeued for subsequent delivery attempts. The command syntax is as follows:

```
jobs [-[no]hosts] [-[no]jobs] [-[no]messages] [channel-name]
```

“[26.3.8.1 Diagnosing and Cleaning up .HELD Messages](#)” on page 854 Rewritten.

Chapter 27, Monitoring Messaging Server

“[27.4.1.2 To Monitor the Size of the Message Queues](#)” on page 873

<use `imsimta qm` and `imsimta summarize`.> [use `imsimta qm` and `imsimta summarize`.]

Appendix A, SNMP Support No substantive changes

Appendix B, Administering Event Notification Service in Messaging Server No substantive changes.

Appendix C, Short Message Service (SMS)

Table C–5

Under Miscellaneous Options:

1) DEBUG - Default: <1>6

2) LISTEN_CONNECTION_MAX - Maximum number of concurrent, inbound TCP connections to allow across all SMPP relay and server instantiations. Default: 10,000

3) LOG_PAGE_COUNT - Controls the value recorded in the mail.log file's message size field to be page count instead of blocks. Default 0

“[C.5.8.3 Miscellaneous](#)” on page 976

New section for LOG_PAGE_COUNT.

Appendix D, Installation Worksheets No differences.

Post-install Tasks and Layout

This chapter assumes that you have read the *Sun Java Communications Suite 5 Deployment Planning Guide* and installed Messaging Server with the Sun Java™ Enterprise System installer. See the *Sun Java Communications Suite 5 Installation Guide*. Performing the following tasks should get you to a point where you have a functioning Messaging Server. You will still want to customize your deployment as well as provision and/or migrate users and groups. Customizing is described in the later chapters of this guide. Provisioning is described in the *Sun Java System Delegated Administrator 6.4 Administration Guide*.

This chapter consists of the following sections:

- “1.1 Creating UNIX System Users and Groups” on page 63
- “1.2 To Prepare Directory Server for Messaging Server Configuration” on page 64
- “1.3 Creating the Initial Messaging Server Runtime Configuration” on page 65
- “1.4 Installing Messaging Server Against a Directory Server Replica” on page 71
- “1.5 Installing Messaging Server Provisioning Tools” on page 72
- “1.6 SMTP Relay Blocking” on page 74
- “1.7 Enabling Startup After a Reboot” on page 76
- “1.8 Handling sendmail Clients” on page 77
- “1.9 Configuring Messenger Express and Communications Express Mail Filters” on page 79
- “1.10 Performance and Tuning” on page 79
- “1.11 Post-Installation Directory Layout” on page 80
- “1.12 Post-Installation Port Numbers” on page 82

1.1 Creating UNIX System Users and Groups

System users run specific server processes, and privileges need to be given to these users so that they have appropriate permissions for the processes they are running.

Set up a system user account and group for all Sun Java System servers, and set permissions for the directories and files owned by that user. To do so, follow the steps below.

Note – For security reasons, in some deployments it may be desirable to have different system administrators for different servers. This is done by creating different system users and groups per server. For example, the system user for Messaging Server would be different from the system user for Web Server, and system administrators Messaging Server would not be able to administer the Web Server.

▼ To Create UNIX System Users and Groups

- 1 **Log in as superuser.**
- 2 **Create a group to which your system users will belong.**
In the following example, the `mail` group is created:
`# groupadd mail`
- 3 **Create the system user and associate it with the group you just created. In addition, set the password for that user.**
In the following example, the user `mailsrv` is created and associated with the `mail` group:
`# useradd -g mail mailsrv`
`useradd` and `usermod` commands are in `/usr/sbin`. See UNIX man pages for more information.
- 4 **You might also need to check the `/etc/group` and `/etc/passwd` files to be sure that the user has been added to the system group that you created.**

Note – Should you decide not to set up UNIX system users and groups prior to installing Messaging Server, you will be able to specify them when you run the “[1.3 Creating the Initial Messaging Server Runtime Configuration](#)” on page [65](#).

1.2 To Prepare Directory Server for Messaging Server Configuration

For complete information on directory preparation and the directory preparation script `comm_dssetup`, see [Chapter 8, “Directory Preparation Tool \(`comm_dssetup.pl`\)” in *Sun Java Communications Suite 5 Installation Guide*](#) in the Installation Guide. This chapter provides instructions on how to run the Directory Server Setup script (`comm_dssetup.pl`) that configures your LDAP Directory Server to work with your Messaging Server, Calendar Server, or Delegated Admin CLI Utility configurations. The `comm_dssetup.pl` script prepares the Directory Server by setting up new schema, index, and data in your Directory Server. It must be

run for new installations of Messaging Server and Communications Express. It is also a good idea to run the latest `comm_dsetup.pl` if you are upgrading any of the component products that depend on Directory Server.

1.3 Creating the Initial Messaging Server Runtime Configuration

The initial runtime configuration program provides a configuration to get your Messaging Server up and running. It is meant to create an *initial runtime configuration* to setup a generic functional messaging server configuration. Thus it gives you a base working configuration from which you can make your specific customizations. The program is only meant to be run once. Subsequent running of this program will result in your configuration being overwritten. To modify your initial runtime configuration, use the configuration utilities described here and in the *Sun Java System Messaging Server 6.3 Administration Reference*.

1.3.1 Messaging Server Prerequisites

Before running the initial runtime configuration program, you must:

- Install and configure the Directory Server. (See the *Sun Java Enterprise System 5 Installation Guide for UNIX*.)
- Run the `comm_dsetup.pl` program. (See “[Messaging Server Postinstallation Configuration](#)” in *Sun Java Communications Suite 5 Installation Guide*.)
- Record your Administration and Directory installation and configuration parameters in the checklists supplied in [Appendix D, “Installation Worksheets.”](#)

1.3.2 Messaging Server Configuration Checklist

When you run the Messaging Server initial runtime configuration program, record your parameters in [Table D–3](#). To answer certain questions, refer to your Directory Server installation checklists in [Appendix D, “Installation Worksheets.”](#)

▼ To Run the Configure Program

This procedure walks you through configuring the Messaging Server initial runtime configuration.

- 1 **Ensure in your setup that DNS is properly configured and that it is clearly specified how to route to hosts that are not on the local subnet.**

- The /etc/defaultrouter should contain the IP address of the gateway system. This address must be on a local subnet.
- The /etc/resolv.conf exists and contains the proper entries for reachable DNS servers and domain suffixes.
- In /etc/nsswitch.conf, the hosts: and ipnodes: line has the files, dns and nis keywords added. The keyword files must precede dns and nis. So if the lines look like this:

```
hosts: nis dns files  
ipnodes: nis dns files
```

They should be changed to this:

```
hosts: files nis dns  
ipnodes: files nis dns
```

- Make sure that the FQDN is the first host name in the /etc/hosts file.

If your Internet host table in your /etc/hosts file looks like this:

```
123.456.78.910 budgie.west.sesta.com  
123.456.78.910 budgie loghost mailhost
```

Change it so that there is only one line for the IP address of the host. Be sure the first host name is a fully qualified domain name. For example:

```
123.456.78.910 budgie.west.sesta.com budgie loghost mailhost
```

- You can verify that the lines are read correctly by running the following commands:

```
# getent hosts ip_address  
# getent ipnodes ip_address
```

If the lines are read correctly, you should see the IP address followed by the FQDN and then the other values. For example:

```
# getent hosts 192.18.126.103  
192.18.126.103 budgie.west.sesta.com budgie loghost mailhost
```

Note – On Solaris OS 10 U3 and earlier platforms, you not only have to add the Fully Qualified Domain Name (FQDN) to the /etc/hosts file, but also to the /etc/inet/ipnodes file. Otherwise, you will get an error indicating that your host name is not a Fully Qualified Domain Name. From Solaris OS 10U4 onwards, the contents of the /etc/inet/ipnodes and /etc/hosts files have been merged together into just the /etc/hosts file. Applying kernel patch 120011-14 on any Solaris 10 system will also perform the merge, and subsequent removal of the /etc/inet/ipnodes file.

2 Invoke the Messaging Server initial runtime configuration with the following command:

`msg-svr-base/sbin/configure [flag]`

You might need to use the `xhost(1)` command if you are configuring Messaging Server on a remote system.

The table below describes optional flags you can set with the `configure` program:

Flag	Description
<code>-nodisplay</code>	Invokes a command-line configuration program.
<code>-noconsole</code>	Invokes a GUI user interface program.
<code>-state [statefile]</code>	Uses a silent installation file. Must be used with <code>-nodisplay</code> and <code>-noconsole</code> flags. See “ To Perform a Silent Installation ” on page 70.

Once you run the `configure` command, the configuration program will start:

3 Welcome

The first panel in the `configure` program is a copyright page. Select Next to continue or Cancel to exit. If you didn’t configure the administration server (Messaging Server 2005Q4 or earlier only) you will be warned, select okay to continue.

4 Enter the Fully Qualified Host Name (FQHN).

This is the machine on which Messaging Server will operate. When you installed the server using the Java Enterprise System installer, you probably specified the physical host name. However, if you are installing a cluster environment, you will want to use the logical hostname. Here is the chance to change what you originally specified.

5 Select directory to store configuration and data files.

Select the directory where you want to store the Messaging Server configuration and data files. Specify a pathname that is not under the `msg-svr-base`. Symbolic links will be created under `msg-svr-base` to the configuration and data directory. For more information on these symbolic links, see “[1.11 Post-Installation Directory Layout](#)” on page 80.

Make sure you have large enough disk space set aside for these files.

6 You will see a small window indicating that components are being loaded.

This may take a few minutes.

7 Select Components to Configure.

Select the Messaging components that you want to configure.

- Message Transfer Agent: Handles routing, delivering user mail, and handling SMTP authentication. The MTA provides support for hosted domains, domain aliases, and server-side filters.
- Message Store: Provides the foundation for unified messaging services through its universal Message Store. Access to the message store is available through multiple protocols (HTTP, POP, IMAP). If you are only configuring a Message Store, you must also select the MTA.
- Webmail Server: Handles the HTTP protocol retrieval of messages from the Message Store. This component is also used by Communication Express to provide web-based access.
- Messaging Multiplexor: Acts as a proxy to multiple messaging server machines within an organization. Users connect to the Multiplexor server, which redirects each connection to the appropriate mail server. This component is not enabled by default. If you do check the MMP as well as the Message Store, they will be enabled on the same system; a warning message will appear for you to change your port numbers after configuration. For instructions on doing so, see “[1.12 Post-Installation Port Numbers](#)” on page 82.

To configure the MMP, see [Chapter 7, “Configuring and Administering Multiplexor Services.”](#)

Check any components you want to configure, and uncheck those components you do not wish to configure.

8 Enter the system user name and the group that will own the configured files.

For information on setting up system users and groups, see “[1.1 Creating UNIX System Users and Groups](#)” on page 63.

9 Configuration Directory Server Panel

Enter your Configuration Directory LDAP URL, Administrator and Password. This is taken from the Administration Server configuration. **Note that this is for Messaging Server 6 2005Q4 and earlier, later versions do not store configuration data in Directory Server and do not use the Administration Server.)**

Gather the Configuration Server LDAP URL from your Directory Server installation. See the Directory Server Installation worksheet from [Table D-1](#).

The Directory Manager has overall administrator privileges on the Directory Server and all Sun Java System servers that make use of the Directory Server (for example, the Messaging Server). It also has full administration access to all entries in the Directory Server. The default and recommended Distinguished Name (DN) is cn=Directory Manager and is set during Directory Server configuration.

Note – If you select something other than the default, you will have a mismatch between the Administration Server and the configuration Directory Server. This will require manual post-configuration steps. So modify this entry only if you really know what you are doing.

10 User/Group Directory Server Panel

Enter your Users and Groups Directory LDAP URL, Administrator and Password.

Gather the User/Group Server LDAP URL information from the host and port number information from your Directory Server installation. See the Directory Server Installation worksheet from [Table D-1](#).

The Directory Manager has overall administrator privileges on the Directory Server and all Sun Java System servers that make use of the Directory Server (for example, the Messaging Server) and has full administration access to all entries in the Directory Server. The default and recommended Distinguished Name (DN) is `cn=Directory Manager` and is set during Directory Server configuration.

If you are installing against a replicated Directory Server instance, you must specify the credentials of the replica, not the master directory.

11 Postmaster Email Address

Enter a Postmaster Email Address.

Select an address that your Administrator will actively monitor. For example, `pma@siroe.com` for a postmaster on the `siroe` domain. This address cannot begin with “Postmaster.”

The user of the email address is not automatically created. Therefore, you will need create it later by using a provisioning tool.

12 Password for administrator accounts

Enter an initial password that will be used for service administrator, server, user/group administrator, end user administrator privileges as well as PAB administrator and SSL passwords.

After the initial runtime configuration, you might change this password for individual administrator accounts. For more information, see [“4.1 To Modify Your Passwords” on page 121](#).

13 Default Email Domain

Enter a Default Email Domain.

This email domain is the default that is used if no other domain is specified. For example, if `siroe.com` is the default email domain, then the domain to which messages addressed to user IDs without a domain will be sent.

If you are using the Delegated Administrator CLI, the command-line interface for provisioning users and groups with Sun LDAP Schema 2, you will want to specify the same default domain during its configuration. For more information, see the [Sun Java System Delegated Administrator 6.4 Administration Guide](#).

14 Organization DN

Enter an Organization DN under which users and groups will be created. The default is the email domain prepended to the user/group suffix.

For example, if your user/group suffix is `o=usergroup`, and your email domain is `siroe.com`, then the default is `o=siroe.com`, `o=usergroup` (where `o=usergroup` is your user/group Directory suffix which was specified in “[1.1 Creating UNIX System Users and Groups](#)” on [page 63](#)).

If you choose the same user/group Directory suffix as your Organization DN, you may have migration problems if you decide to create a hosted domain. If you want to set up a hosted domain during initial runtime configuration, then specify a DN one level below the User/Group suffix.

15 Ready to Configure

The configuration program will check for enough disk space on your machine and then outline the components it is ready to configure.

To configure the Messaging components, select Configure Now. To change any of your configuration variables, select Back. Or to exit from the configuration program, select Cancel.

16 Starting Task Sequence, Sequence Completed, and Installation Summary Panels

You can read the installation status by selecting Details on the final Installation Summary page. To exit the program, select Close.

A log file is created in `msg-svr-base/install/configure_YYYYMMDDHHMMSS.log`, where `YYYYMMDDHHMMSS` identifies the 4-digit year, month, date, hour, minute, and second of the configuration.

An initial runtime configuration is now set up for your Messaging Server. To change any configuration parameter, refer to other parts of this document for instructions on doing so.

To start Messaging Server, use the following command:

`/opt/SUNWmsgsr/sbin/start-msg`

▼ To Perform a Silent Installation

The Messaging Server initial runtime configuration program automatically creates a silent installation *state* file (called `saveState`) that can be used to quickly configure additional Messaging Server instances in your deployment where the Messaging Server Solaris packages have been installed. All of your responses to the configuration prompts are recorded in that file.

By running the silent installation, you instruct the `configure` program to read the silent installation state file. The `configure` program uses the responses in this file rather than ask the same installation questions again for subsequent initial runtime configurations of Messaging

Server. When you use the state file in a new installation, you are not asked any questions. Instead, all of the state file responses are automatically applied as the new installation parameters.

The silent installation `saveState` *state* file is stored in the `msg-svr-base/install/configure_YYYYMMDDHHMMSS` directory, where `YYYYMMDDHHMMSS` identifies the 4-digit year, month, date, hour, minute, and second of the `saveState` file.

To use the silent installation *state* file to configure another Messaging Server instance on another machine in the deployment, follow these steps:

1 Copy the silent installation *state* file to a temporary area on the machine where you are performing the new installation.

2 Review and edit the silent installation *state* file as necessary.

You will probably want to change some of the parameters and specifications in the *state* file. For example, the default email domain for the new installation may be different than the default email domain recorded in the *state* file. Remember that the parameters listed in the *state* file will be automatically applied to this installation.

3 Run the following command to configure other machines with the silent installation file:

```
msg-svr-base/sbin/configure -nodisplay -noconsole -state \  
fullpath/saveState
```

where *fullpath* is the full directory path of where the `saveState` file is located. (See Step 1 of this section).

Note – After running the silent installation program, a new *state* file is created from the silent installation in directory location:

`msg-svr-base/install/configure_YYYYMMDDHHMMSS/saveState`, where `YYYYMMDDHHMMSS` identifies the 4-digit year, month, date, hour, minute, and second of the directory containing the `saveState` file.

1.4 Installing Messaging Server Against a Directory Server Replica

There might be limitations that prevent you from installing Messaging Server against a Directory Server master:

- You do not have Directory Server master credentials.
- Messaging Server cannot communicate directly with the Directory Server master.

▼ To Install Messaging Server Against a Directory Server Replica

- 1 Run the `comm_dsetup.pl` program against all Directory Servers including the Directory Server replicas (see “[Messaging Server Postinstallation Configuration](#)” in *Sun Java Communications Suite 5 Installation Guide*).
- 2 Run the **Messaging configure** program using the replicated Directory Server credentials as described in “[1.3 Creating the Initial Messaging Server Runtime Configuration](#)” on page 65.
By default, this program is located in `msg-svr-base/sbin/configure`.
Because of invalid privileges, the `configure` program will fail in trying to configure the Directory Server Administrators. It will, however, produce the `msg-svr-base/config/*.ldif` files that are needed to allow proper privileges to the Directory Server replicas.
- 3 Move the `*.ldif` files to the Directory Server master.
- 4 Run the `ldapmodify` command on the `*.ldif` files.
See the Sun Java System Directory Server documentation for more information on `ldapmodify` or in the `msg-svr-base/install/configure_YYYYMMDDHHMMSS.log`.
- 5 Run the `configure` program again.
Your Directory Server replica (and master) are now configured to work with your Messaging Server.

1.5 Installing Messaging Server Provisioning Tools

The following sections provide a summary of install information about the supported provisioning tools:

- “[1.5.1 Schema 1 Delegated Administrator for Messaging](#)” on page 72
- “[1.5.2 LDAP Provisioning Tools](#)” on page 74
- “[1.1 Creating UNIX System Users and Groups](#)” on page 63

1.5.1 Schema 1 Delegated Administrator for Messaging

Two GUI provisioning tools are available for Messaging Server, the iPlanet Delegated Administrator (Sun LDAP Schema 1) and the Communications Services Delegated Administrator (Sun LDAP Schema 2). This section discusses the former. For details on the latter see the *Sun Java System Delegated Administrator 6.4 Administration Guide*.

To install the iPlanet Delegated Administrator (Sun LDAP Schema 1), you need to download it from the Sun Software page. Contact your Sun Java System representative for information on the download location information.

Note – The iPlanet Delegated Administrator can only be installed after Messaging Server and Web Server are installed and configured. For more information on installing iPlanet Delegated Administrator, see the iPlanet Delegated Administrator documentation.

iPlanet Delegated Administrator is only available for those customers with existing Messaging Server 5.x installations and who are currently installing Messaging Server 6. It is not available to those customers new to the Messaging Server product.

iPlanet Delegated Administrator must be used with Sun Java System Web Server 6.0 (which is only bundled with the previous Messaging Server 5.2 product). You cannot use Web Server 6.1 (bundled with the Java Enterprise System installer) with iPlanet Delegated Administrator.

Note – When you install the following products, use the Java Enterprise System installer. Note that some of these products have their own configuration whereas other product configurators are embedded in the Java Enterprise System installer/configurator. For more information, refer to specific product documentation.

▼ To Install iPlanet Delegated Administrator

1 Be sure that Sun Java System Directory Server 5.2 is installed and configured.

For more information, read the appropriate *Sun Java System Directory Server Installation Guide*.

2 Install and configure Messaging Server.

Messaging Server will detect that you are using Sun LDAP Schema 1 since Sun Java System Access Manager will not be installed.

3 Install Sun Java System Web Server 6.0 from your previous Messaging Server 5.2 bundle.

Review the Sun Java System Web Server documentation and the Sun Java System Delegated Administrator documentation.

4 Install iPlanet Delegated Administrator for Messaging 1.2 Patch 2.

Contact your Sun support representative to obtain the latest version.

Refer to the iPlanet Delegated Administrator documentation.

1.5.2 LDAP Provisioning Tools

Sun LDAP Schema 1 users and groups can be provisioned using the LDAP Directory tools (Schema 2 is not supported).

▼ To Install Schema 1 LDAP Provisioning Tools

- 1 **If Directory Server is not already installed, be sure to install and configure it.**

For more information, refer to the *Sun Java Enterprise System 5 Installation Guide for UNIX*.

- 2 **Configure Access Manager to recognize data in your Directory Server.**

Before Access Manager can recognize the data in your LDAP directory, you must add special object classes to entries for all organizations, groups and users that will be managed by Access Manager. If you have not done this already, do it before you start provisioning new accounts. Sample scripts are bundled in the Access Manager product to help you automatically add these object classes to your directory. For more information on these post-installation steps, see the *Sun Java System Access Manager Migration Guide*.

- 3 **Install and configure Messaging Server with help from this guide.**

Messaging Server will detect which Sun Java System LDAP Schema you are using, depending on whether or not Access Manager is installed.

- 4 **Install and configure Sun Java System Web Server 6.1 to enable mail filtering in Messenger Express.**

For more information on enabling mail filtering, see “[1.9 Configuring Messenger Express and Communications Express Mail Filters](#)” on page 79.

Though mail filtering is not a provisioning tool, its functionality existed in the previous GUI version of Delegated Administrator for Messaging.

- 5 **Refer to the Sun Java System Messaging Server documentation to perform LDAP provisioning.**

For Sun LDAP Schema 1 LDAP provisioning, use the *iPlanet Messaging Server 5.2 Provisioning Guide* and *Sun Java Communications Suite 5 Schema Reference*. The Schema Reference contains object classes and attributes for both Sun LDAP Schema 1 and v.2.

1.6 SMTP Relay Blocking

By default, Messaging Server is configured to block attempted SMTP relays; that is, it rejects attempted message submissions to external addresses from unauthenticated external sources (external systems are any other system than the host on which the server itself resides). This default configuration is quite aggressive in blocking SMTP relaying in that it considers all other systems to be external systems.

After installation, it is important to manually modify your configuration to match the needs of your site. Specifically, your messaging server should recognize its own internal systems and subnets from which SMTP relaying should always be accepted. If you do not update this configuration, you might encounter problems when testing your MTA configuration.

IMAP and POP clients that attempt to submit messages via Messaging Server system's SMTP server destined for external addresses, and who do not authenticate using SMTP AUTH (SASL), will find their submission attempts rejected. Which systems and subnets are recognized as internal is typically controlled by the INTERNAL_IP mapping table, which may be found in the file *msg-svr-base/config/mappings*.

For instance, on a Messaging Server system whose IP address is 192.45.67.89, the default INTERNAL_IP mapping table would appear as follows:

INTERNAL_IP

```
$(192.45.67.89/32) $Y  
127.0.0.1 $Y  
* $N
```

The initial entry, using the \$(IP-pattern/significant-prefix-bits) syntax, is specifying that any IP address that matches the full 32 bits of 192.45.67.89 should match and be considered internal. The second entry recognizes the loopback IP address 127.0.0.1 as internal. The final entry specifies that all other IP addresses should not be considered internal.

You may add additional entries by specifying additional IP addresses or subnets before the final \$N entry. These entries must specify an IP address or subnet (using the \$(.../...) syntax to specify a subnet) on the left side and \$Y on the right side. Or you may modify the existing \$(.../...) entry to accept a more general subnet.

For instance, if this same sample site has a class C network, that is, it owns all of the 192.45.67.0 subnet, then the site would want to modify the initial entry so that the mapping table appears as follows:

INTERNAL_IP

```
$(192.45.67.89/24) $Y  
127.0.0.1 $Y  
* $N
```

Or if the site owns only those IP addresses in the range 192.45.67.80-192.45.67.99, then the site would want to use:

INTERNAL_IP

```
! Match IP addresses in the range 192.45.67.80-192.45.67.95  
$(192.45.67.80/28) $Y
```

```
! Match IP addresses in the range 192.45.67.96-192.45.67.99
$(192.45.67.96/30) $Y
127.0.0.1 $Y
* $N
```

Note that the *msg-svr-base/sbin/imsimta test-match* utility can be useful for checking whether an IP address matches a particular \$(. . . / . . .) test condition. The *imsimta test-mapping* utility can be more generally useful in checking that your INTERNAL_IP mapping table returns the desired results for various IP address inputs.

After modifying your INTERNAL_IP mapping table, be sure to issue the *msg-svr-base/sbin/imsimta cnbuild* and the *msg-svr-base/sbin/imsimta restart* utilities so that the changes take effect.

Further information on the mapping file and general mapping table format, as well as information on *imsimta* command line utilities, can be found in [Chapter 2, “Message Transfer Agent Command-line Utilities,” in Sun Java System Messaging Server 6.3 Administration Reference](#). In addition, information on the INTERNAL_IP mapping table can be found in [“18.6 To Add SMTP Relaying” on page 557](#).

1.7 Enabling Startup After a Reboot

You can enable Messaging Server startup after system reboots by using the bootup script: *msg-svr-base/lib/Sun_MsgSvr*. That is, by default, Messaging Server will not restart after a system reboot unless you run this script. In addition, this script can also start up your MMP, if enabled.

▼ To Enable Messaging Server After a Reboot

- 1 Copy the *Sun_MsgSvr* script into the */etc/init.d* directory.
- 2 Change the following ownerships and access modes of the *Sun_MsgSvr* script:

Ownership (chown(1M))	Group Ownership (chgrp(1M))	Access Mode (chmod(1M))
root (superuser)	sys	0744

- 3 Go to the */etc/rc2.d* directory and create the following link:

```
ln /etc/init.d/Sun_MsgSvr S92Sun_MsgSvr
```

- 4 Go to the */etc/rc0.d* directory and create the following link:

```
ln /etc/init.d/Sun_MsgSvr K08Sun_MsgSvr
```

1.8 Handling sendmail Clients

If end users send messages through sendmail clients, you can configure Messaging Server to work with those clients over protocol. Users can continue to use the UNIX sendmail client.

To create compatibility between sendmail clients and Messaging Server, you can create and modify a `sendmail` configuration file.

Note – Each time a new sendmail patch is applied to your system, you will need to modify the `submit.cf` file as described in the following instructions in “[To Create the sendmail Configuration File on Solaris 9 Platforms](#)” on page 78. On Solaris 8, follow the instructions in “[To Obtain the Proper Version of the /usr/lib/sendmail on Solaris 8](#)” on page 77.

When you installed previous versions of Messaging Server, the `/usr/lib/sendmail` binary was replaced with a component of the Messaging Server product. In Messaging Server 6.0 to the current version, this replacement during install is no longer necessary. Therefore, you may need to obtain the proper version of the `/usr/lib/sendmail` binary from the most current sendmail patch.

On Solaris OS 9 platforms, `sendmail` is no longer a setuid program. Instead, it is a setgid program.

▼ To Obtain the Proper Version of the /usr/lib/sendmail on Solaris 8

- 1 Find the file `main-v7sun.mc` file in directory `/usr/lib/mail/cf` and create a copy of this file.

In the example in this section, a copy called `sunone-msg.mc` is created.

- 2 In the `sunone-msg.mc` file, add the following lines before the MAILER macros:

```
FEATURE('nullclient', 'smtp:rhino.west.sesta.com')dnl  
MASQUERADE_AS('west.sesta.com')dnl  
define('confDOMAIN_NAME', 'west.sesta.com')dnl
```

`rhino.west.sesta.com` is the localhost name and `west.sesta.com` is the default email domain as described in “[1.3 Creating the Initial Messaging Server Runtime Configuration](#)” on page 65. In an HA environment, use the logical host name. See [Chapter 3, “Configuring High Availability,”](#) for more information about logical hostnames for High Availability.

- 3 Compile the `sunone-msg.mc` file:

```
/usr/ccs/bin/make sunone-msg.cf
```

The `sunone-msg.mc` will output `sunone-msg.cf`.

- 4 Make a backup copy of the existing `sendmail.cf` file located in the `/etc/mail` directory.
 - a. Copy and rename `/usr/lib/mail/cf/sunone-msg.cf` to `sendmail.cf` file.
 - b. Move the new `sendmail.cf` file to the `/etc/mail` directory.

▼ To Create the sendmail Configuration File on Solaris 9 Platforms

- 1 Find the file `submit.mc` file in directory `/usr/lib/mail/cf` and create a copy of this file.
In the example in this section, a copy called `sunone-submit.mc` is created.

- 2 Change the following line in the file `sunone-submit.mc`:

```
FEATURE("msp")dn
```

to

```
FEATURE("msp", "rhino.west.sesta.com")dnl
```

where `rhino.west.sesta.com` is the localhost name.

`rhino.west.sesta.com` is the localhost name and `west.sesta.com` is the default email domain as described in “[1.3 Creating the Initial Messaging Server Runtime Configuration](#)” on page 65. In an HA environment, use the logical host name. See [Chapter 3, “Configuring High Availability,”](#) for more information about logical hostnames for High Availability.

- 3 Compile the `sunone-submit.mc` file:

```
/usr/ccs/bin/make sunone-submit.cf
```

The `sunone-submit.mc` will output `sunone-submit.cf`.

- 4 Make a backup copy of the existing `submit.cf` file in the `/etc/mail` directory.

- a. Copy and rename `/usr/lib/mail/cf/sunone-submit.cf` file to `submit.cf` file.

- b. Move the new `submit.cf` file to the `/etc/mail` directory.

1.9 Configuring Messenger Express and Communications Express Mail Filters

Mail filters are accessible through Messenger Express and Communications Express. There is no need to deploy the .war file if you use only Communications Express, but to deploy the mail filters within Messenger Express you need to issue the following commands:

If you're using Web Server as your web container:

```
# cd web_svr_base/bin/https/httpadmin/bin
# ./wdeploy deploy -u /MailFilter -i https-srvr_instance \
-v https-virtual_srvr_instance msg_svr_base/SUNWmsgmf/MailFilter.war
```

If using Application Server as your Web container:

```
# cd app_svr_base/sbin
# ./asadmin
asadmin> deploy --user admin msg_svr_base/SUNWmsgmf/MailFilter.war
```

In both cases, set the following configutil parameter and restart mshttpd:

```
# cd msg_svr_base/sbin
# ./configutil -o "local.webmail.sieve.port" \
-v "WS_port_no|AS_port_no"
# ./stop-msg http
# ./start-msg http
```

Information on mail filters for end-users is available in the Messenger Express and Communications Express online help files.

1.10 Performance and Tuning

This section provides some post-installed performance and tuning information. For more complete information on this topic, refer to “[Performance Considerations for a Messaging Server Architecture](#)” in *Sun Java Communications Suite 5 Deployment Planning Guide*.

1.10.1 Java Message Queue (JMQ) Tuning

A configuration parameter in JMQ that may need tweaking in a typical JMQ deployment. Change the following line in the file

/usr/share/lib/imq/props/broker/default.properties from:

```
# imq.autocreate.destination.maxNumProducers=100
```

to `imq.autocreate.destination.maxNumProducers=-1`.

If you don't do this, you may see a message in the JMQ log that mentions `Destination Conflicts`, and you may lose notifications on large system deployments.

1.11 Post-Installation Directory Layout

After installing the Sun Java System Messaging Server, its directories and files are arranged in the organization described in [Table 1–1](#). The table is not exhaustive; it shows only those directories and files of most interest for typical server administration tasks.

TABLE 1–1 Post-Installation Directories and Files

Directory	Default Location and Description
Messaging Server Base <i>(msg_svr_base)</i>	/opt/SUNWmsgsr/ (default location) The directory on the Messaging Server machine dedicated to holding the server program, configuration, maintenance, and information files. Only one Messaging Server Base directory per machine is permitted.
Configuration <i>config</i>	<i>msg_svr_base/config/</i> Contains all of the Messaging Server configuration files such as the <code>imta.cnf</code> and the <code>msg.conf</code> files. On Solaris and Linux platforms only: This directory is symbolically linked (on UNIX platforms) to the <code>config</code> subdirectory of the data and configuration directory (default: <code>/var/opt/SUNWmsgsr/</code>) that you specified in the initial runtime configuration.
Log <i>log</i>	<i>msg_svr_base/log/</i> Contains the Messaging Server log files like the <code>mail.log_current</code> file. On Solaris and Linux platforms only: This directory is symbolically linked (on UNIX platforms) to the <code>log</code> subdirectory of the data and configuration directory (default: <code>/var/opt/SUNWmsgsr/</code>) that you specified in the initial runtime configuration.

TABLE 1-1 Post-Installation Directories and Files *(Continued)*

Directory	Default Location and Description
Data	<i>msg_svr_base/data/</i>
data	(required location) Contains databases, configuration, log files, site-programs, queues, store and message files. The data directory includes the config and log directories.
System Administrator Programs	On Solaris and Linux platforms only: This directory is symbolically linked (on UNIX platforms) to the data and configuration directory (default: /var/opt/SUNWmsgsr/) that you specified in the initial runtime configuration.
sbin	<i>msg_svr_base/sbin/</i> (required location) Contains the Messaging Server system administrator executable programs and scripts such as imsmta, configutil, stop-msg, start-msg, and uninstaller.
Library	<i>msg_svr_base/lib/</i>
lib	(required location) Contains shared libraries, private executable programs and scripts, daemons, and non-customizable content data files. For example: imapd and qm_maint.hlp.
SDK Include Files	<i>msg_svr_base/include/</i>
include	(required location) Contains Messaging header files for Software Development Kits (SDK).
Examples	<i>msg_svr_base/examples/</i>
examples	(required location) Contains the examples for various SDKs, such as Messenger Express AUTH SDK.
Installation Data	<i>msg_svr_base/install/</i>
install	(required location) Contains installation-related data files such as installation log files, silent installation files, factory default configuration files, and the initial runtime configuration log files.

1.12 Post-Installation Port Numbers

In the installation and initial runtime configuration programs, port numbers will be chosen for various services. These port numbers can be any number from 1 to 65535.

[Table 1–2](#) lists the port numbers that are designated after installation.

TABLE 1–2 Port Numbers Designated During Installation

Port Number	Service (<code>configutil</code> parameter)
389	Standard Directory Server LDAP Port on the machine where you install Directory Server. This port is specified in the Directory Server installation program. (<code>local.ugldapport</code>)
110	Standard POP3 Port. This port may conflict with the MMP port if installed on the same machine. (<code>service.pop.port</code>)
143	Standard IMAP4 Port. This port may conflict with the MMP port if installed on the same machine. (<code>service imap port</code>)
25	Standard SMTP Port. (<code>service.http.smtpport</code>)
80	Messenger Express HTTP Port. This port may conflict with the Web Server port if installed on the same machine. (<code>service.http.port</code>)
995	POP3 over SSL port. For encrypted communications. (<code>service.pop.sslport</code>)
993	IMAP over SSL Port. For encrypted communications. This port may conflict with the MMP port if installed on the same machine. (<code>service imap.sslport</code>)
443	HTTP over SSL Port. For encrypted communications. (<code>service.http.sslport</code>)
7997	Messaging and Collaboration Event Notification Service (ENS) Port.
27442	Port that is used by the Job Controller for internal product communication.
49994	Port that is used by the Watcher for internal product communication. See the <i>Sun Java System Messaging Server Administration Guide</i> for more information on the Watcher. (<code>local.watcher.port</code>)

If certain products are installed on the same machine, you will encounter port number conflicts. [Table 1–3](#) shows potential port number conflicts.

TABLE 1–3 Potential Port Number Conflicts

Conflicting Port Number	Port	Conflicting Port
995	POP3 over SSL	MMP POP3 Proxy with SSL
143	IMAP Server	MMP IMAP Proxy

TABLE 1-3 Potential Port Number Conflicts *(Continued)*

Conflicting Port Number	Port	Conflicting Port
110	POP3 Server	MMP POP3 Proxy
993	IMAP over SSL	MMP IMAP Proxy with SSL
80	Web Server port	Messenger Express

If possible, you should install products with conflicting port numbers on separate machines. If you are unable to do so, then you will need to change the port number of one of the conflicting products.

▼ To Change Port Numbers

- Use the `configutil` utility to change port numbers.

See “[configutil](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for complete syntax and usage.

Example 1-1 Changing the Messenger Express HTTP Port Number

The following example uses the `service.http.port` configutil parameter to change the Messenger Express HTTP port number to 8080.

```
# configutil -o service.http.port -v 8080
# stop-msg http
$start-msg http
```


Upgrading from Messaging Server 5.2 to Sun Java System Messaging Server

The information in this chapter has been moved to a technical article entitled [Upgrading from Messaging Server 5.2 to Sun Java System Messaging Server](#) (<http://www.sun.com/bigadmin/hubs/comms/files/ms52-ms6-upgrade.html>). See this document and the *Sun Java Communications Suite 5 Upgrade Guide* for complete information. Note that “20.15 Migrating or Moving Mailboxes to a New System” on page 675 is still in this book.

2.1 Information Moved

See [Upgrading from Messaging Server 5.2 to Sun Java System Messaging Server](#) (<http://www.sun.com/bigadmin/hubs/comms/files/ms52-ms6-upgrade.html>) and *Sun Java Communications Suite 5 Upgrade Guide*.

Configuring High Availability

This section provides the information you need to configure the Veritas Cluster Server or Sun Cluster high availability clustering software and prepare it for use with the Messaging Server. It is assumed you have read [Chapter 6, “Designing for Service Availability,” in *Sun Java Communications Suite 5 Deployment Planning Guide*](#) as well as the appropriate Veritas or Sun Cluster Server documentation for detailed planning, installation instructions, required patches, and other information as needed.

This chapter consists of the following sections:

- [“3.1 Supported Versions” on page 87](#)
- [“3.2 High Availability Models” on page 87](#)
- [“3.3 Installing Messaging Server High Availability—Overview” on page 94](#)
- [“3.4 Sun Cluster Installation” on page 95](#)
- [“3.5 Veritas Cluster Server Agent Installation” on page 116](#)
- [“3.6 Unconfiguring High Availability” on page 119](#)

3.1 Supported Versions

See [“What’s New in This Release of Messaging Server” in *Sun Java Communications Suite 5 Release Notes*](#) for the latest supported versions and platforms.

3.2 High Availability Models

There are different high availability models that can be used with Messaging Server. Three of the more basic ones are:

- [“3.2.1 Asymmetric” on page 88](#)
- [“3.2.2 Symmetric” on page 89](#)
- [“3.2.3 N+1 \(N Over 1\)” on page 91](#)
- [“3.2.4 Choosing a High Availability Model” on page 93](#)

- “3.2.5 System Down Time Calculations” on page 93

Each of these models is described in greater detail in the following subsections.

Note that different HA products may or may not support different models. Refer to the HA documentation to determine which models are supported.

3.2.1 Asymmetric

The basic asymmetric or *hot standby* high availability model consists of two clustered host machines or *nodes*. A logical IP address and associated host name are designated to both nodes.

In this model, only one node is active at any given time; the backup or hot standby node remains idle most of the time. A single shared disk array between both nodes is configured and is mastered by the active or *primary* node. The message store partitions and Mail Transport Agent (MTA) queues reside on this shared volume.

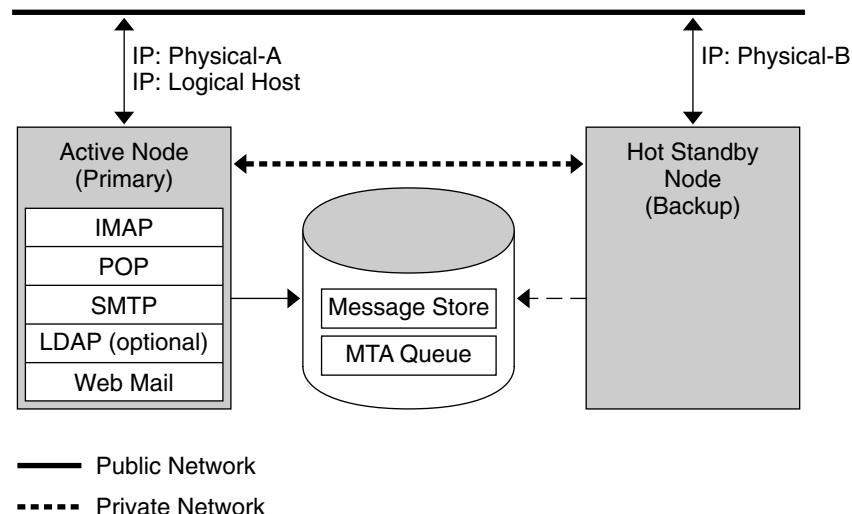


FIGURE 3-1 Asymmetric High Availability Mode

The preceding figure shows two physical nodes, Physical-A and Physical-B. Before failover, the active node is Physical-A. Upon failover, Physical-B becomes the active node and the shared volume is switched so that it is mastered by Physical-B. All services are stopped on Physical-A and started on Physical-B.

The advantage of this model is that the backup node is dedicated and completely reserved for the primary node. Additionally, there is no resource contention on the backup node when a

failover occurs. However, this model also means that the backup node stays idle most of the time and this resource is therefore under utilized.

3.2.2 Symmetric

The basic symmetric or "dual services" high availability model consists of two hosting machines, each with its own logical IP address. Each logical node is associated with one physical node, and each physical node controls one disk array with two storage volumes. One volume is used for its local message store partitions and MTA queues, and the other is a mirror image of its partner's message store partitions and MTA queues.

The following figure shows the symmetric high availability mode. Both nodes are active concurrently, and each node serves as a backup node for the other. Under normal conditions, each node runs only one instance of Messaging Server.

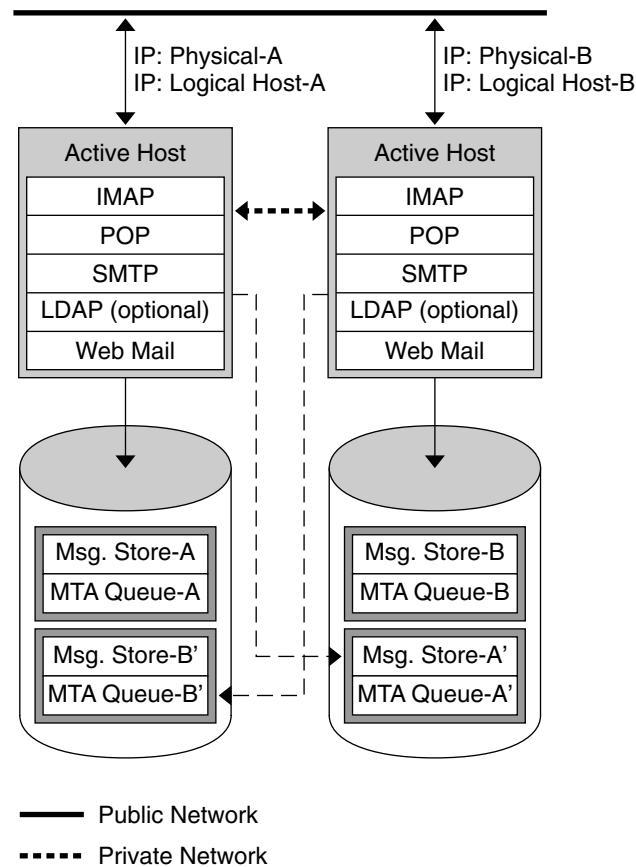


FIGURE 3–2 Symmetric High Availability Mode

Upon failover, the services on the failing node are shut down and restarted on the backup node. At this point, the backup node is running Messaging Server for both nodes and is managing two separate volumes.

The advantage of this model is that both nodes are active simultaneously, thus fully utilizing machine resources. However, during a failure, the backup node will have more resource contention as it runs services for Messaging Server from both nodes. Therefore, you should repair the failed node as quickly as possible and switch the servers back to their dual services state.

This model also provides a backup storage array. In the event of a disk array failure, its redundant image can be picked up by the service on its backup node.

To configure a symmetric model, you need to install shared binaries on your shared disk. Note that doing so might prevent you from performing rolling upgrades, a feature that enables you to update your system during Messaging Server patch releases. (This feature is planned for future releases.)

3.2.3 N+1 (N Over 1)

The N + 1 or "N over 1" model operates in a multi-node asymmetrical configuration. N logical host names and N shared disk arrays are required. A single backup node is reserved as a hot standby for all the other nodes. The backup node is capable of concurrently running Messaging Server from the N nodes.

The figure below illustrates the basic N + 1 high availability model.

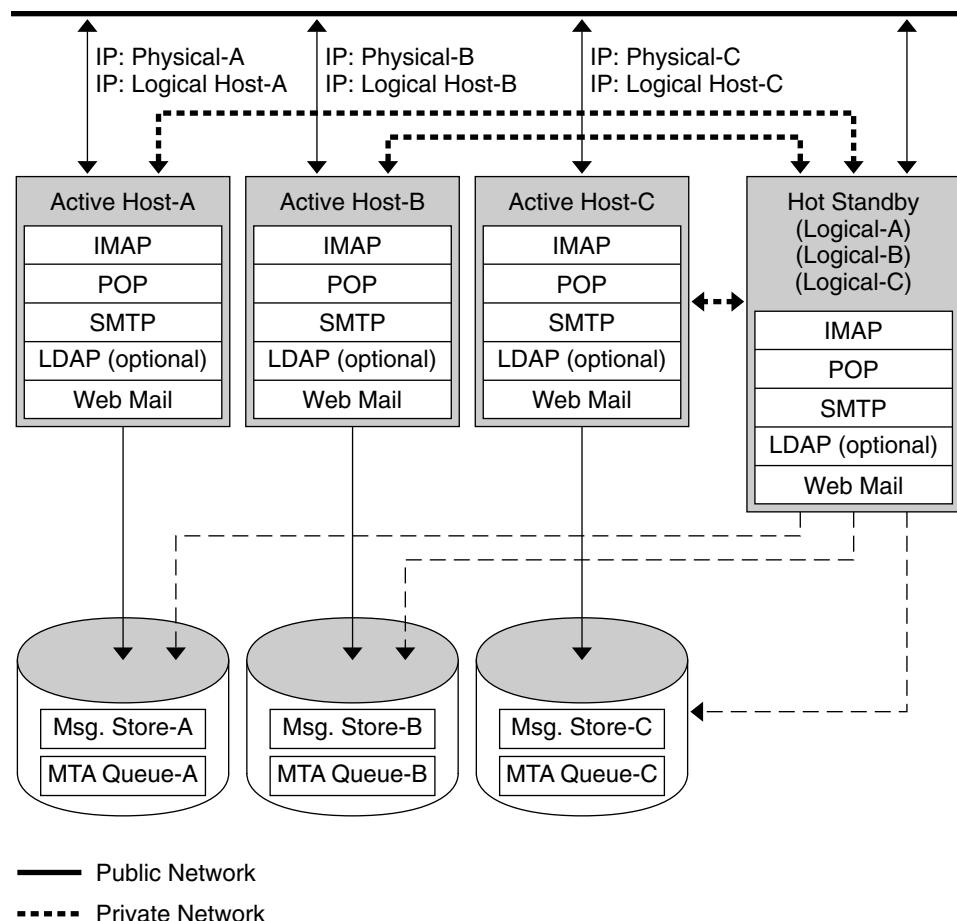


FIGURE 3–3 N + 1 High Availability Mode

Upon failover of one or more active nodes, the backup node picks up the failing node's responsibilities.

The advantages of the $N + 1$ model are that the server load can be distributed to multiple nodes and that only one backup node is necessary to sustain all the possible node failures. Thus, the machine idle ratio is $1/N$ as opposed to $1/1$, as is the case in a single asymmetric model.

To configure an $N+1$ model, you need to install binaries only on the local disks (that is, not shared disks as with the symmetric model). The current Messaging Server installation and setup process forces you to put the binaries on the shared disk for any symmetric, 1+1, or $N+1$ asymmetrical or symmetrical HA solution.

3.2.4

Choosing a High Availability Model

The following table summarizes the advantages and disadvantages of each high availability model. Use this information to help you determine which model is right for your deployment.

TABLE 3-1 HA Model Comparison

Model	Advantages	Disadvantages	Recommended Users
Asymmetric	<ul style="list-style-type: none"> ■ Simple Configuration ■ Backup node is 100 percent reserved 	Machine resources are not fully utilized.	A small service provider with plans to expand in the future
Symmetric	<ul style="list-style-type: none"> ■ Better use of system resources ■ Higher availability 	Resource contention on the backup node. HA requires fully redundant disks.	A small corporate deployment that can accept performance penalties in the event of a single server failure
N + 1	<ul style="list-style-type: none"> ■ Load distribution ■ Easy expansion 	Management and configuration complexity.	A large service provider who requires distribution with no resource constraints

3.2.5

System Down Time Calculations

The following table illustrates the probability that on any given day the messaging service will be unavailable due to system failure. These calculations assume that on average, each server goes down for one day every three months due to either a system crash or server hang, and that each storage device goes down one day every 12 months. These calculations also ignore the small probability of both nodes being down simultaneously.

TABLE 3-2 HA Down Probabilities

Model	Server Down Time Probability
Single server (no high availability)	$\text{Pr}(\text{down}) = (4 \text{ days of system down} + 1 \text{ day of storage down})/365 = 1.37\%$
Asymmetric	$\text{Pr}(\text{down}) = (0 \text{ days of system down} + 1 \text{ day of storage down})/365 = 0.27\%$
Symmetric	$\text{Pr}(\text{down}) = (0 \text{ days of system down} + 0 \text{ days of storage down})/365 = (\text{near } 0)$
N + 1 Asymmetric	$\text{Pr}(\text{down}) = (5 \text{ hours of system down} + 1 \text{ day of storage down})/(365xN) = 0.27\%/N$

3.3 Installing Messaging Server High Availability—Overview

After selecting the appropriate HA model for your deployment, you'll want to choose between the Sun Cluster HA or the Veritas HA. This section provides preliminary HA deployment information. Subsequent sections will provide specific information on Sun Cluster and Veritas High Availability solutions.

3.3.1 Cluster Agent Installation

A cluster agent is a Messaging Server program that runs under the cluster framework.

The Sun Cluster Messaging Server agent (`SUNWscims`) is installed when you select Sun Cluster through the Java Enterprise System installer. The Veritas Cluster Messaging Server agent (`SUNWmsgvc`) can be found in the Messaging Server Product subdirectory on the Sun Java Communications Suite CD, `Solaris_sparc/Product/messaging_svr/Packages/SUNWmsgvc`. (Note that you must use the `pkgadd(1M)` command to install the VCS cluster agent.)

3.3.2 Messaging Server and High Availability Notes

Some items of note regarding the Messaging Server and high availability (applies to both Veritas Cluster and Sun Cluster) installation:

- High availability for the Messaging Server is not installed by default; be sure to select High Availability Components from the Custom Installation menu of the Java Enterprise System installer.
- When running the installation, make sure that the HA logical host name and associated IP addresses for Messaging Servers are functioning (for example, active). The reason for this is because portions of the installation will make TCP connections using them. Run the installation on the cluster node currently pointed at by the HA logical host name for the messaging server.
- Make sure that the `msg_svr_base` is on the shared file system; otherwise, high availability will not work correctly. For example, after failing over to another node, the servers will no longer see the data accumulated by the servers on the failed node.
- When you are asked for the fully-qualified domain name of the Messaging Server host during initial runtime configuration, be sure to specify the fully-qualified HA logical host name for the Messaging Server. During the install, TCP connections using this logical host name will be attempted.
- When you are asked for the IP address of Messaging Server when you run `ha_ip_config`, be sure to specify the IP address associated with the logical host name for Messaging Server. Do not use the IP address for the physical host.

- Clustering software needs to be installed before installing and configuring the current version of Messaging Server. Run the installation on the cluster node currently pointed at by the HA logical host name for Messaging Server. Use the cluster alias when prompted for any node names.
- When running the Messaging Server Initial Runtime Configuration (see “[1.3 Creating the Initial Messaging Server Runtime Configuration](#)” on page 65) sure to specify the fully-qualified HA logical host name of the cluster of Messaging Server.
- Configure Messaging Server using the cluster host name. If you do otherwise, then you’ll need to re-configure a second time using the cluster host name.

3.3.3

Using the useconfig Utility

The `useconfig` utility allows you to share a single configuration between multiple nodes in an HA environment. This utility is not meant to upgrade or update an existing configuration.

For example, if you are upgrading your first node, you will install through the Communications Suite Installer and then configure Messaging Server. You will then failover to the second node where you will install the Messaging Server package through the Communications Suite Installer, but you will not have to run the Initial Runtime Configuration Program (`configure`) again. Instead, you can use the `useconfig` utility.

To enable the utility, run `useconfig` to point to your previous Messaging Server configuration:

```
msg-svr-base/sbin/useconfig install/configure_YYYYMMDDHHMMSS
```

where `configure_YYYYMMDDHHMMSS` is your previous configuration settings file.

On a brand new node, you can find the `configure_YYYYMMDDHHMMSS` in the `msg-svr-base/data/setup` directory on the shared disk.

The following sections on “[3.5 Veritas Cluster Server Agent Installation](#)” on page 116 and “[3.4 Sun Cluster Installation](#)” on page 95 describe when you can use the `useconfig` utility.

3.4 Sun Cluster Installation

This section describes how to install and configure the Messaging Server as a Sun Cluster Highly Available (HA) Data Service. The following topics are covered in this section:

- “[3.4.1 Sun Cluster Requirements](#)” on page 96
- “[3.4.2 About HAStoragePlus](#)” on page 96
- “[3.4.3 Configuring Messaging Server with Sun Cluster HAStorage or HAStoragePlus](#)” on page 97
- “[3.4.4 Binding IP Addresses on a Server](#)” on page 114

See also Sun Cluster documentation.

Note that Veritas File System (VxFS) is supported with Sun Cluster 3.1.

3.4.1 Sun Cluster Requirements

This section presumes the following:

- Sun Cluster is installed and configured on Solaris operating system with required patches.
- The Sun Cluster agent SUNWscims is installed on your systems.
- If logical volumes are being created, either Solstice DiskSuite or Veritas Volume Manager is used.

3.4.2 About HAStoragePlus

It is highly recommended that you use the HAStoragePlus resource type to make locally mounted file systems highly available within a Sun Cluster environment. Local file systems, also called Failover File Systems (FFS) provide better Input/Output performance than cluster file systems (CFS), also called global file systems. HAStoragePlus supports both FFS and CFS. In contrast, HAStorage only supports CFS.

HAStoragePlus has a number of benefits:

- HAStoragePlus bypasses the global file service layer completely. For disk-IO intensive data services, this leads to a significant performance increase.
- HAStoragePlus can work with any file system (like UFS, VxFS, and so forth), even those that might not work with the global file service layer. If a file system is supported by the Solaris operating system, it will work with HAStoragePlus.

To determine whether to create HAStorage or HAStoragePlus resources in a data service resource group, consider the following criteria:

- Use HAStoragePlus if you are using Sun Cluster 3.0 Release May 2002 or Sun Cluster 3.1
- Use HAStorage if you are using Sun Cluster 3.0 December 2001 or earlier

For more information on HAStoragePlus, refer to appropriate Sun Cluster docs. For example, <http://docs.sun.com/app/docs/coll/573.10>.

3.4.3 Configuring Messaging Server with Sun Cluster HAStorage or HAStoragePlus

This section describes how to configure HAStorage and HAStoragePlus for Messaging Server for Sun Cluster. The first section describes the general steps. Subsequent sections show specific examples for symmetric and asymmetric deployments.

After configuring HA, be sure to review “[3.4.4 Binding IP Addresses on a Server](#)” on page 114 for additional steps associated with HA support.

The following description assumes that Messaging Server has been configured with an HA logical host name and IP address. The physical host names are assumed to be `mars` and `venus`, with an HA logical host name of `meadow`. [Figure 3–4](#) depicts the nested dependencies of the different HA resources you will create in configuring Messaging Server HA support.

Note – While we describe how to configure HAStorage and HAStoragePlus, we highly recommend HAStoragePlus better I/O performance. See “[3.4.2 About HAStoragePlus](#)” on page 96.

This section contains the following subsections:

- “[To Configure Messaging Server with Sun Cluster HAStorage or HAStoragePlus—Generic Example](#)” on page 98
- “[To Unconfigure Messaging Server HA Support for Sun Cluster 3.x—Generic Example](#)” on page 103
- “[To Configure a Two-node Symmetric Messaging Server—Example](#)” on page 104
- “[To Unconfigure an HA Symmetric Deployment](#)” on page 109
- “[To Configure a Two-node HA Asymmetric Messaging Server—Example](#)” on page 110
- “[3.4.3.1 How to Enable Debugging on Sun Cluster](#)” on page 113

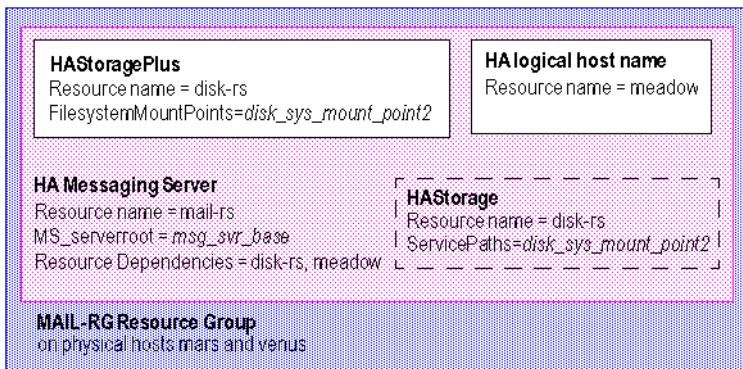


FIGURE 3–4 A Simple Messaging ServerHA Configuration

▼ To Configure Messaging Server with Sun Cluster HAStorage or HAStoragePlus—Generic Example

This section provides the generic steps for configuring Messaging Server for HA. After reviewing these steps, refer to the specific asymmetric or symmetric examples in the following sections. In these instructions the physical hosts are called `mars` and `venus`. The logical host name is `meadow`.

Figure 3–4 depicts the nested dependencies of the different HA resources you will create in configuring Messaging Server HA support.

1 Become the superuser and open a console.

All of the following Sun Cluster commands require that you have logged in as superuser. You will also want to have a console or window for viewing messages output to `/dev/console`.

2 On all the nodes, install the required Messaging Sun Cluster Data Service Agents package (SUNWscims).

3 On each node in the cluster create the Messaging Server runtime user and group under which the Messaging Server will run.

The user ID and group ID numbers must be the same on all nodes in the cluster. The *runtime user ID* is the user name under which Messaging Server runs. This name should not be root. The default is `mailsrv`. The *runtime Group ID* is the group under which Messaging Server runs. The default is `mail`.

Although the `configure` utility can create these names for you, you can also create them before running `configure` as part of the preparation of each node as described in this chapter. The runtime user and group ID names must be in the following files:

- `mailsrv`, or the name you select, must be in `/etc/passwd` on all nodes in the cluster
- `mail`, or the name you select, must be in `/etc/group` on all nodes in the cluster

See “[1.1 Creating UNIX System Users and Groups](#)” on page 63.

4 Add required resource types to Sun Cluster.

Configure Sun Cluster to know about the resources types we will be using. To register Messaging Server as your resource use the following command:

```
# scrgadm -a -t SUNW.ims
```

To register HAStoragePlus as a resource type, use this command:

```
# scrgadm -a -t SUNW.HAStoragePlus
```

To do the same with HAStorage as a resource type, use this command:

```
# scrgadm -a -t SUNW.HAStorage
```

5 Create a failover resource group for the Messaging Server.

If you have not done so already, create a resource group and make it visible on the cluster nodes which will run the Messaging Server. The following command creates a resource group named MAIL-RG, making it visible on the cluster nodes mars and venus:

```
# scrgadm -a -g MAIL-RG -h mars,venus
```

You may, of course, use whatever name you wish for the resource group.

6 Create an HA logical host name resource and bring it on-line.

If you have not done so, create and enable a resource for the HA logical host name placing that resource in the resource group. The following command does so using the logical host name meadow. Since the -j switch is omitted, the name of the resource created will also be meadow. meadow is the logical host name by which clients communicate with the services in the resource group.

```
# scrgadm -a -L -g MAIL-RG -l meadow
# scswitch -Z -g MAIL-RG
```

7 Create an HAStorage or HAStoragePlus resource.

Next, you need to create an HA Storage or HAStoragePlus resource type for the file systems on which Messaging Server is dependent. The following command creates an HAStoragePlus resource named disk-rs, and the file system *disk_sys_mount_point* is placed under its control:

```
# scrgadm -a -j disk-rs -g MAIL-RG \
-t SUNW.HAStoragePlus \
-x FilesystemMountPoints=disk_sys_mount_point-1, disk_sys_mount_point-2 -x AffinityOn=True
```

SUNW.HAStoragePlus represents the device groups, cluster and local file systems which are to be used by one or more data service resources. One adds a resource of type SUNW.HAStoragePlus to a resource group and sets up dependencies between other resources and the SUNW.HAStoragePlus resource. These dependencies ensure that the data service resources are brought online after:

- All specified device services are available (and collocated if necessary)
- All specified file systems are mounted following their checks

The `FilesystemMountPoints` extension property allows for the specification of either global or local file systems. That is, file systems that are either accessible from all nodes of a cluster or from a single cluster node. Local file systems managed by a `SUNW.HAStoragePlus` resource are mounted on a single cluster node and require the underlying devices to be Sun Cluster global devices. `SUNW.HAStoragePlus` resources specifying local file systems can only belong in a failover resource group with affinity switch overs enabled. These local file systems can therefore be termed failover file systems. Both local and global file system mount points can be specified together.

A file system whose mount point is present in the `FilesystemMountPoints` extension property is assumed to be local if its `/etc/vfstab` entry satisfies both of the following conditions:

- Non-global mount option
- Mount at boot flag is set to no

Note – Instances of the `SUNW.HAStoragePlus` resource type ignore the mount at boot flag for global file systems.

For the `HAStoragePlus` resource, the comma-separated list of `FilesystemMountPoints` are the mount points of the Cluster File Systems (CFS) or Failover File Systems (FFS) on which Messaging Server is dependent. In the above example, only two mount points, `disk_sys_mount_point-1` and `disk_sys_mount_point-2`, are specified. If one of the servers has additional file systems on which it is dependent, then you can create an additional HA storage resource and indicate this additional dependency in [Step 15](#).

For `HAStorage` use the following:

```
# scrgadm -a -j disk-rs -g MAIL-RG \
-t SUNW.HAStorage
-x ServicePaths=disk_sys_mount_point-1, disk_sys_mount_point-2 -x AffinityOn=True
```

For the `HAStorage` resource, the comma-separated list of `ServicePaths` are the mount points of the cluster file systems on which Messaging Server is dependent. In the above example, only two mount points, `disk_sys_mount_point-1` and `disk_sys_mount_point-2`, are specified. If one of the servers has additional file systems on which it is dependent, then you can create an additional HA storage resource and indicate this additional dependency in [Step 15](#).

8 Install the required Messaging Server packages on the primary node. Choose the *Configure Later* option.

Use the Communications Suite installer to install the Messaging Server packages.

For symmetric deployments: Install Messaging Server binaries and configuration data on files systems mounted on a shared disk of the Sun Cluster. For example, for Messaging Server

binaries could be under /disk_sys_mount_point-1/SUNWmsgsr and the configuration data could be under /disk_sys_mount_point-2/config.

For asymmetric deployments: Install Messaging Server binaries on local file systems on each node of the Sun Cluster. Install configuration data on a shared disk. For example, the configuration data could be under /disk_sys_mount_point-2/config.

9 Configure the Messaging Server. See “[1.3 Creating the Initial Messaging Server Runtime Configuration](#)” on page 65.

In the initial runtime configuration, you are asked for the Fully Qualified Host Name. You must use the HA logical hostname instead of the physical hostname.

In the initial runtime configuration, you are asked to specify a configuration directory in “[1.3 Creating the Initial Messaging Server Runtime Configuration](#)” on page 65. Be sure to use the shared disk directory path of your HAStorage or HAStoragePlus resource.

10 Run the ha_ip_config script to set service.listenaddr and service.http.smtphost and to configure the dispatcher.cnf and job_controller.cnf files for high availability.

The script ensures that the logical IP address is set for these parameters and files, rather than the physical IP address. It also enables the watcher process (sets local.watcher.enable to 1), and auto restart process (local.autorestart to 1).

For instructions on running the script, see “[3.4.4 Binding IP Addresses on a Server](#)” on page 114.

The ha_ip_config script should only be run once on the primary node.

11 Modify the imta.cnf file and replace all occurrences of the physical hostname with the logical hostname of the cluster.

12 Fail the resource group from the primary to the secondary cluster node in order to ensure that the failover works properly.

Manually fail the resource group over to another cluster node. (Be sure you have superuser privileges on the node to which you failover.)

Use the scstat command to see what node the resource group is currently running on (“online” on). For instance, if it is online on mars, then fail it over to venus with the command:

```
# scswitch -z -g MAIL-RG -h venus
```

If you are upgrading your first node, you will install through the Communications Suite Installer and then configure Messaging Server. You will then failover to the second node where you will install the Messaging Server package through the Communications Suite Installer, but you will not have to run the Initial Runtime Configuration Program (configure) again. Instead, you can use the useconfig utility.

- 13 Install the required Messaging Server packages on the secondary node. Choose the *Configure Later* option.**

After failing over to the second node, install the Messaging Server packages using the Communications Suite Installer.

For symmetric deployments: Do not install Messaging Server.

For asymmetric deployments: Install Messaging Server binaries on local file systems on the local file system.

- 14 Run useconfig on the second node of the cluster.**

The useconfig utility allows you to share a single configuration between multiple nodes in an HA environment. You don't need to run the initial runtime configure program (*configure*). Instead use the useconfig utility (see “[3.3.3 Using the useconfig Utility](#)” on page [95](#)).

- 15 Create an HA Messaging Server resource.**

It's now time to create the HA Messaging Server resource and add it to the resource group. This resource is dependent upon the HA logical host name and HA disk resource.

In creating the HA Messaging Server resource, we need to indicate the path to the Messaging Server top-level directory—the *msg-svr-base* path. These are done with the *IMS_serverroot* extension properties as shown in the following command.

```
# scrgadm -a -j mail-rs -t SUNW.ims -g MAIL-RG \
-x IMS_serverroot=msg-svr-base \
-y Resource_dependencies=disk-rs,meadow
```

The above command, creates an HA Messaging Server resource named *mail-rs* for the Messaging Server, which is installed on *IMS_serverroot* in the *msg-svr-base* directory. The HA Messaging Server resource is dependent upon the HA disk resource *disk-rs* as well as the HA logical host name *meadow*.

If the Messaging Server has additional file system dependencies, then you can create an additional HA storage resource for those file systems. Be sure to include that additional HA storage resource name in the *Resource_dependencies* option of the above command.

- 16 Enable the Messaging Server resource.**

It's now time to activate the HA Messaging Server resource, thereby bringing the messaging server online. To do this, use the command

```
# sccswitch -e -j mail-rs
```

The above command enables the *mail-rs* resource of the *MAIL-RG* resource group. Since the *MAIL-RG* resource was previously brought online, the above command also brings *mail-rs* online.

- 17 Verify that things are working.**

Use the *scstat -pv* command to see if the *MAIL-RG* resource group is online.

You may also want to look at the output directed to the console device for any diagnostic information. Also look in the `syslog` file, `/var/adm/messages`. For more debugging options and information, refer to “[3.4.3.1 How to Enable Debugging on Sun Cluster](#)” on page 113.

▼ To Unconfigure Messaging Server HA Support for Sun Cluster 3.x—Generic Example

This section describes how to undo the HA configuration for Sun Cluster. This section assumes the simple example configuration (described in the “[3.4 Sun Cluster Installation](#)” on page 95 (for example, [Step 3](#)) may be different but will otherwise follow the same logical order.

1 Become the superuser.

All of the following Sun Cluster commands require that you be running as user superuser.

2 Bring the resource group offline.

To shut down all of the resources in the resource group, issue the command

```
# scswitch -F -g MAIL-RG
```

This shuts down all resources within the resource group (for example, the Messaging Server and the HA logical host name).

3 Disable the individual resources.

Next, remove the resources one-by-one from the resource group with the commands:

```
# scswitch -n -j mail-rs
# scswitch -n -j disk-rs
# scswitch -n -j budgie
```

4 Remove the individual resources from the resource group.

Once the resources have been disabled, you may remove them one-by-one from the resource group with the commands:

```
# scrgadm -r -j mail-rs
# scrgadm -r -j disk-rs
# scrgadm -r -j budgie
```

5 Remove the resource group.

Once the all the resources have been removed from the resource group, the resource group itself may be removed with the command:

```
# scrgadm -r -g MAIL-RG
```

6 Remove the resource types (optional).

Should you need to remove the resource types from the cluster, issue the commands:

```
# scrgadm -r -t SUNW.ims  
# scrgadm -r -t SUNW.HASstoragePlus
```

▼ To Configure a Two-node Symmetric Messaging Server—Example

In this example we assume two cluster nodes with the physical hostnames `mars.red.siroe.com` and `venus.red.siroe.com`. The installation and configuration directory locations need to be unique. A contention problem will occur if the installation and configuration directories on each node have the same directory names, for example `/opt/SUNWmsgs` and `/var/opt/SUNWmsgs`. The contention problem occurs when `venus` does a failover to `mars`, and two instances of Messaging Server compete with the same install and configuration directories.

A good practice for creating unique names for the installation and configuration directories would be to use the format `/opt/NodeMember/SUNWmsgs` for the installation directory and `/var/opt/NodeMember/SUNWmsgs` for the configuration directory. You can use any directory to install your binaries and configuration data as long as they are unique.

In this example we assume two cluster nodes with physical hostnames `mars.red.siroe.com` and `venus.red.siroe.com`.

For `mars.red.siroe.com`, binaries are installed at `/opt/mars/SUNWmsgs` and configuration data is installed at `/var/opt/mars/SUNWmsgs`.

For `venus.red.siroe.com`, binaries are installed at `/opt/venus/SUNWmsgs` and configuration data is installed at `/var/opt/venus/SUNWmsgs`.

We have two logical hostnames called `meadow` and `pasture` with their respective logical IP addresses. For example, the `/etc/hosts` file on both nodes look like this:

```
192.18.75.155 meadow.red.siroe.com meadow  
192.18.75.157 pasture.red.siroe.com pasture
```

1 Install the Messaging Server Sun Cluster agent package (SUNWscims) on both nodes.**2 Create four file systems.**

These files systems can either be Cluster File Systems or local file systems (Failover File Systems).

```
/var/opt/mars/SUNWmsgs  
/var/opt/venus/SUNWmsgs  
/opt/mars/SUNWmsgs  
/opt/venus/SUNWmsgs
```

These file systems should be mounted on a shared disk. For example below we show four Cluster File Systems. The contents of the /etc/vfstab shown below should be similar on all nodes of the cluster.

```
# cat /etc/vfstab
#device device mount FS fsck mount mount to mount to fsck point type
pass at_boot_options
/dev/md/penguin/dsk/d500 /dev/md/penguin/rdsk/d500 /opt/mars/SUNWmsgsr ufs 2 yes
logging,global
/dev/md/penguin/dsk/d400 /dev/md/penguin/rdsk/d400 /var/opt/mars/SUNWmsgsr ufs 2
yes logging,global
/dev/md/polarbear/dsk/d200 /dev/md/polarbear/rdsk/d200 /opt/venus/SUNWmsgsr ufs 2
yes logging,global
/dev/md/polarbear/dsk/d300 /dev/md/polarbear/rdsk/d300 /var/opt/venus/SUNWmsgsr
ufs 2 yes logging,global
```

If you want to make the four file systems shown above as local file systems (Fail Over File Systems), set the mount at boot option to no and remove the mount option `global` keyword:

3 Configure the primary node

a. Add the required resource types on the primary node.

This configures Sun Cluster to know about the resources types that will be used. To register Messaging Server and the HAStoragePlus resource, use the following commands:

```
# scrgadm -a -t SUNW.HAStoragePlus
# scrgadm -a -t SUNW.ims
```

b. Create a fail over resource group for Messaging Server called MS_RG_MARS.

```
# scrgadm -a -g MS_RG_MARS -h mars,venus
```

c. Create one logical hostname resource called meadow, add it to the resource group and bring it on-line.

```
# scrgadm -a -L -g MS_RG_MARS -l meadow
# scrgadm -c -j meadow -y R_description="LogicalHostname resource for meadow"
# scswitch -Z -g MS_RG_MARS
```

d. Create a HAStoragePlus resource called ms-haspmars with the file systems created earlier.

```
# scrgadm -a -j ms-haspmars -g MS_RG_MARS -t SUNW.HAStoragePlus -x
FileSystemMountPoints ="/opt/mars/SUNWmsgsr, /var/opt/mars/SUNWmsgsr" -x
AffinityOn=TRUE
```

e. Enable the HAStoragePlus resource:

```
# scswitch -e -j ms-haspmars
```

4 Install the Messaging Server on the primary node.

Install the Messaging Server packages using Communications Suite installer. Make sure you install the Messaging Server binaries and configuration data on the shared file system (see [Step 2](#)). For example, for this instance of Messaging Server, the messaging binaries are under /opt/mars/SUNWmsgsr and the configuration data is under /var/opt/mars/SUNWmsgsr.

5 Install and configure the Messaging Server on the primary node (see “[1.3 Creating the Initial Messaging Server Runtime Configuration](#)” on page 65).

The initial runtime configuration program asks for the Fully Qualified Host Name. Enter the logical hostname meadow.red.siroe.com. The program also asks to specify a configuration directory. Enter /var/opt/mars/SUNWmsgsr.

6 Run the ha_ip_config script on the primary node and provide the logical IP address.

This is only run on the primary node and not on the secondary node. The ha_ip_config script is located under the installation directory under the sbin directory. For example:

```
# /opt/mars/SUNWmsgsr/sbin/ha_ip_config

Please specify the IP address assigned to the HA logical host name.
Use dotted decimal form, a.b.c.d

Logical IP address: 192.18.75.155
# This value is the logical IP address of the logical hostname. Refer
# to the /etc/hosts file.
Please specify the path to the top level directory in which iMS is
installed.

iMS server root: /opt/mars/SUNWmsgsr

.

.

Updating the file /opt/mars/SUNWmsgsr/config/dispatcher.cnf
Updating the file /opt/mars/SUNWmsgsr/config/job_controller.cnf
Setting the service.listenaddr configutil parameter
Setting the local.snmp.listenaddr configutil parameter
Setting the service.http.smtphost configutil parameter
Setting the local.watcher.enable configutil parameter
Setting the local.autorestart configutil parameter
Setting the metermaid.config.bindaddr configutil parameters
Setting the metermaid.config.serveraddr configutil parameters
Setting the local.ens.port parameter
Configuration successfully updated
```

7 Modify the imta.cnf file and replace all occurrences of the physical hostname, that is, mars, with the HA logical host name (meadow).

8 Fail over the resource group to the secondary node (venus).

After failing over, you will then configure the secondary node (venus).

```
# scswitch -z -g MS_RG_VENUS -h mars
```

9 On the secondary node (venus) run useconfig utility. See “[3.3.3 Using the useconfig Utility](#)” on page [95](#)

You do not have to run the initial runtime configuration program (*configure*) or install the Messaging Server packages.

In the following example, */var/opt/mars/SUNWmsgsr* is the shared configuration directory.

```
# useconfig /var/opt/mars/SUNWmsgsr/setup/configure_20061201124116
cp /var/opt/mars/SUNWmsgsr/setup/configure_20061201124116/Devsetup.properties
/opt/mars/SUNWmsgsr/lib/config-templates/Devsetup.properties
/usr/sbin/groupadd mail
/usr/sbin/useradd -g mail -d / mailsrv
/usr/sbin/usermod -G mail mailsrv
sed -e "s/local.serveruid/mailsrv/" -e "s/local.serveruid/mail/" -e "s:<msg·RootPath>:/opt/mars/SUNWmsgsr:" \
/opt/mars/SUNWmsgsr/lib/config-templates/devtypes.txt.template >
/opt/mars/SUNWmsgsr/lib/config-templates/devtypes.txt
sed -e "s/local.serveruid/mailsrv/" -e "s/local.serveruid/mail/" -e
"s:<msg·RootPath>:/opt/mars/SUNWmsgsr:"
/opt/mars/SUNWmsgsr/lib/config-templates/config.ins.template >
/opt/mars/SUNWmsgsr/lib/config-templates/config.ins
/opt/mars/SUNWmsgsr/lib/devinstall -l sepadmsvr:pkgcfg:config -v -m -i
/opt/mars/SUNWmsgsr/lib/config-templates/config.ins
/opt/mars/SUNWmsgsr/lib/config-templates
/opt/mars/SUNWmsgsr/lib/jars /opt/mars/SUNWmsgsr/lib
devinstall returned 0
crle -c /var/ld/ld.config -s
/usr/lib/secure:/opt/SUNWmsgsr/lib:/opt/venus/SUNWmsgsr/lib:/opt/mars/SUNWmsgsr/lib
-s /opt/mars/SUNWmsgsr/lib
See /opt/mars/SUNWmsgsr/install/useconfiglog_20061211155037 for more details
```

10 Create the HA Messaging Server Resource and enable it.

```
# scrgadm -a -j ms-rs-mars -t SUNW.ims -g MS_RG_MARS -x IMS_serverroot
=:/opt/mars/SUNWmsgsr -y Resource_dependencies=meadow,ms-haspmars
# scswitch -e -j mail-rs-mars
```

The above command, creates an HA Messaging Server resource named *ms-rs-mars* for the Messaging Server, which is installed on */opt/mars/SUNWmsgsr*. This HA Messaging Server resource is dependent upon the HA disk resource, that is, the file systems created earlier as well as the HA logical host name *meadow*.

11 Verify that everything is working.

Failover the Messaging Server resource back to the primary node.

```
# scswitch -z -g MAIL-RG -h mars
```

- 12 Similarly, create another failover resource group for the second instance of Messaging Server with venus as the primary and mars as the secondary (or standby node).**

Repeat the steps from 3 to 10 with venus as the primary node for this resource group, MS_RG_VENUS as the resource group, pasture as the logical hostname and ms-hasp-venus as the HAStoragePlus resource. Thus the commands will look like this:

To create the resource group MS_RG_VENUS:

```
# scrgadm -a -g MS_RG_VENUS -h venus,mars
```

To create a logical hostname resource called pasture, add it to the resource group and bring it on-line;

```
# scrgadm -a -L -g MS_RG_VENUS -l pasture
# scrgadm -c -j pasture -y R_description="LogicalHostname resource for pasture"
# scswitch -Z -g MS_RG_VENUS
```

To create an HAStoragePlus resource called ms-hasp-venus with the file systems created earlier:

```
# scrgadm -a -j ms-hasp-venus -g MS_RG_VENUS -t SUNW.HAStoragePlus -x
FileSystemMountPoints ="/opt/venus/SUNWmsgsr, /var/opt/venus/SUNWmsgsr" -x
AffinityOn=TRUE
```

To enable the HAStoragePlus resource:

```
# scswitch -e -j ms-hasp-venus
```

To run the ha_ip_config script on the primary node and provide the logical IP address:

```
# /opt/venus/SUNWmsgsr/sbin/ha_ip_config
```

To create the HA Messaging Server Resource and enable it:

```
# scrgadm -a -j ms-rs-venus -t SUNW.ims -g MS_RG_VENUS -x IMS_serverroot
=/opt/venus/SUNWmsgsr -y Resource_dependencies=pasture,ms-hasp-venus
# scswitch -e -j mail-rs-venus
```

To fail over the resource group to the secondary node (venus):

```
# scswitch -z -g MS_RG_MARS -h venus
```

To run useconfig on the secondary node (mars) run useconfig utility:

```
# useconfig /var/opt/venus/SUNWmsgsr/setup/configure_20061201124116
```

To verify that everything is working by failing over the Messaging Server resource back to the primary node:

```
# scswitch -z -g MAIL-RG -h venus
```

▼ To Unconfigure an HA Symmetric Deployment

Unconfiguring is done when you need to upgrade the Messaging Server or the Sun Cluster, or when you need to uninstall the Messaging Server. It is assumed that the system was configured the using the previous example.

The first step is to remove each resource group in the cluster. In the example, there are two resource groups, MS_RG_MARS and MS_RG_VENUS. Both must be removed.

1 Remove resource group MS_RG_MARS from the cluster.

Use the following commands on just one node. You do not have to do this on each node.

a. Brings the resource group off-line on all nodes of the cluster:

```
# scswitch -F -g MS_RG_MARS
```

b. Disable all the specific Messaging Server resources:

```
# scswitch -n -j ms-rs-mars  
# scswitch -n -j meadow  
# scswitch -n -j ms-haspmars
```

c. Remove all the specific MS resources:

```
# scrgadm -r -j ms-rs-mars  
# scrgadm -r -j meadow  
# scrgadm -r -j ms-haspmars
```

d. Remove the resource group:

```
scrgadm -r -g MS_RG_MARS
```

2 Remove resource group MS_RG_VENUS from the cluster.

Use the following commands on just one node. You do not have to do this on each node.

a. Brings the resource group off-line on all nodes of the cluster:

```
# scswitch -F -g MS_RG_VENUS
```

b. Disable all the specific Messaging Server resources:

```
# scswitch -n -j ms-rs-venus  
# scswitch -n -j pasture  
# scswitch -n -j ms-haspmenus
```

c. Remove all the specific MS resources:

```
# scrgadm -r -j ms-rs-venus  
# scrgadm -r -j pasture  
# scrgadm -r -j ms-haspmenus
```

d. Remove the resource group:

```
scrgadm -r -g MS_RG_VENUS
```

3 Unregister the Resource Types that are not in use.

```
# scrgadm -r -t SUNW.HAStoragePlus
# scrgadm -r -t SUNW.ims
```

▼ To Configure a Two-node HA Asymmetric Messaging Server—Example

In this example we assume two node cluster with physical hostnames `daisy.red.siroe.com` and `Lavender.red.siroe.com` with a logical hostname called `budgie`.

For `daisy.red.siroe.com`, binaries are installed at `/opt/SUNWmsgsr` and configuration data is installed at `/var/opt/SUNWmsgsr`.

The logical hostname `budgie` is assigned logical IP address. For example, the `/etc/hosts` file could look like this:

```
192.18.75.157 budgie.red.siroe.com budgie
```

1 Install the Messaging Sun Cluster agents (SUNWscims) on both nodes.

2 Create the file system.

In this example, the file system `/var/opt/SUNWmsgsr` is mounted on a shared disk. This file system can either be a Cluster File System or local file systems (Failover File Systems).

3 Configure the primary node (daisy).

a. Add the required resource types on the primary node.

This configures Sun Cluster to know about the resources types that will be used. To register Messaging Server and the HAStoragePlus resource, use the following commands:

```
# scrgadm -a -t SUNW.HAStoragePlus
# scrgadm -a -t SUNW.ims
```

b. Create a resource group for the Messaging Server instance called `MS_RG_DAISY`.

```
# scrgadm -a -g MS_RG_daisy -h daisy,lavender
```

c. Create a logical hostname resource called `meadow`, add it to the resource group and bring it on-line.

```
# scrgadm -a -L -g MS_RG_DAISY -l meadow
# scrgadm -c -j meadow -y R_description="LogicalHostname resource for meadow"
# scswitch -Z -g MS_RG_DAISY
```

- d. Create an HAStoragePlus resource called `ms-hasdp-daisy` with the file systems created earlier.

```
# scrgadm -a -j ms-hasdp-daisy -g MS_RG_DAISY -t SUNW.HAStoragePlus -x
FileSystemMountPoints ="/var/opt/SUNWmsgsr" -x
AffinityOn=TRUE
```

- e. Enable the HAStoragePlus resource:

```
# scswitch -e -j ms-hasdp-daisy
```

- 4 Install and configure the Messaging Server on the primary node (see “[1.3 Creating the Initial Messaging Server Runtime Configuration](#)” on page 65).

The initial runtime configuration program asks for the Fully Qualified Host Name. Enter the logical hostname `meadow.red.siroe.com`. The program also asks to specify a configuration directory. Enter `/var/opt/SUNWmsgsr`.

- 5 Run the `ha_ip_config` script on the primary node and provide the logical IP address.

This is only run on the primary node and not on the secondary node. The `ha_ip_config` script is located under the installation directory under the `sbin` directory. For Example:

```
# /opt/SUNWmsgsr/sbin/ha_ip_config
```

```
Please specify the IP address assigned to the HA logical host name.
Use dotted decimal form, a.b.c.d
```

```
Logical IP address: 192.18.75.155
```

```
# This value is the logical IP address of the logical hostname. Refer
# to the /etc/hosts file.
```

```
Please specify the path to the top level directory in which iMS is
installed.
```

```
iMS server root: /opt/SUNWmsgsr
```

```
...
```

```
Updating the file /opt/SUNWmsgsr/config/dispatcher.cnf
```

```
Updating the file /opt/SUNWmsgsr/config/job_controller.cnf
```

```
Setting the service.listenaddr configutil parameter
```

```
Setting the local.snmp.listenaddr configutil parameter
```

```
Setting the service.http.smtphost configutil parameter
```

```
Setting the local.watcher.enable configutil parameter
```

```
Setting the local.autorestart configutil parameter
```

```
Setting the metermaid.config.bindaddr configutil parameters
```

```
Setting the metermaid.config.serveraddr configutil parameters
```

```
Setting the local.ens.port parameter
```

```
Configuration successfully updated
```

- 6 Modify the `imta.cnf` file and replace all occurrences of the physical hostname (`daisy`) with the HA logical host name (`meadow`).**

- 7 Fail over the resource group to the secondary node (`lavender`).**

After failing over, you will then configure the secondary node (`lavender`).

```
# scswitch -z -g MS_RG_LAVENDER -h daisy
```

- 8 On the secondary node (`lavender`) install Messaging Server and run the `useconfig` utility. See “[3.3.3 Using the `useconfig` Utility](#)” on page 95**

You do not have to run the initial runtime configuration program (`configure`).

In the following example, `/var/opt/SUNWmsgsr` is the shared configuration directory.

```
# useconfig /var/opt/SUNWmsgsr/setup/configure_20061201124116
cp /var/opt/SUNWmsgsr/setup/configure_20061201124116/Devsetup.properties
/opt/SUNWmsgsr/lib/config-templates/Devsetup.properties
/usr/sbin/groupadd mail
/usr/sbin/useradd -g mail -d / mailsrv
/usr/sbin/usermod -G mail mailsrv
sed -e "s/local.serveruid/mailsrv/" -e "s/local.serveruid/mail/" -e "s:<msg·RootPath>:/opt/SUNWmsgsr:"
/opt/SUNWmsgsr/lib/config-templates/devtypes.txt.template >
/opt/SUNWmsgsr/lib/config-templates/devtypes.txt
sed -e "s/local.serveruid/mailsrv/" -e "s/local.serveruid/mail/" -e
"s:<msg·RootPath>:/opt/SUNWmsgsr:"
/opt//SUNWmsgsr/lib/config-templates/config.ins.template >
/opt/SUNWmsgsr/lib/config-templates/config.ins
/opt/SUNWmsgsr/lib/devinstall -l sepadmsvr:pkgcfg:config -v -m -i
/opt/SUNWmsgsr/lib/config-templates/config.ins
/opt/SUNWmsgsr/lib/config-templates
/opt/SUNWmsgsr/lib/jars /opt/SUNWmsgsr/lib
devinstall returned 0
crle -c /var/ld/ld.config -s
/usr/lib/secure:/opt/SUNWmsgsr/lib:/opt/SUNWmsgsr/lib:/opt/SUNWmsgsr/lib
-s /opt/SUNWmsgsr/lib
See /opt/SUNWmsgsr/install/useconfiglog_20061211155037 for more details
```

- 9 Create the HA Messaging Server Resource and enable it.**

```
# scrgadm -a -j ms-rs-daisy -t SUNW.ims -g MS_RG_DAISY -x IMS_serverroot
=:/opt/SUNWmsgsr -y Resource_dependencies=meadow,ms-hasps-daisy
# scswitch -e -j mail-rs-daisy
```

The above command creates an HAMessaging Server resource named `ms-rs-daisy` for the Messaging Server, which is installed on `/opt/SUNWmsgsr`. This HAMessaging Server resource is dependent upon the HA disk resource, that is, the file system created earlier as well as the HA logical host name `meadow`.

10 Verify that everything is working.

Failover the Messaging Server resource back to the primary node.

```
# scswitch -z -g MAIL-RG -h daisy
```

3.4.3.1

How to Enable Debugging on Sun Cluster

Messaging Server Data Service Sun Cluster agents uses two APIs to log debug messages:

`scds_syslog_debug()` writes a debugging message to the system log at 1 level.

`scds_syslog()` writes a message to the system log at `daemon.notice`, `daemon.info` and `daemon.error` levels.

All `syslog` messages are prefixed with the following:

```
SC[resourceTypeName, resourceGroupName, resourceName, methodName]
```

For example:

```
Dec 11 18:24:46 mars SC[SUNW.ims,MS-RG,mail-rs,ims_svc_start]: [ID 831728daemon.debug]
Groupname mail exists.
Dec 11 18:24:46 mars SC[SUNW.ims,MS-RG,mail-rs,ims_svc_start]: [ID 383726daemon.debug]
Username mailsrv exists.
Dec 11 18:24:46 mars SC[SUNW.ims,MS-RG,mail-rs,ims_svc_start]: [ID 244341daemon.debug]
IMS_serverroot = /opt/mars/SUNWmsgsr
Dec 11 15:55:52 mars SC[SUNW.ims,MS_RG,MessagingResource,ims_svc_validate]:
[ID 855581daemon.error] Failed to get the configuration info
Dec 11 18:24:46 mars SC[SUNW.ims,MS-RG,mail-rs,ims_svc_start]: [ID 833212daemon.info]
Attempting to start the data service under process monitor facility.
```

To log messages from the Messaging Server Resource Type `SUNW.ims`, create the Resource Type Directory under `/var/cluster` as shown below:

```
mkdir -p /var/cluster/rgm/rt/SUNW.ims
```

To see all debugging messages for resource type `SUNW.ims`, issue the following command on all the nodes of your cluster:

```
echo 9 > /var/cluster/rgm/rt/SUNW.ims/loglevel
```

To suppress debugging messages for resource type `SUNW.iws`, issue the following command on all nodes of your cluster:

```
echo 0 > /var/cluster/rgm/rt/SUNW.ims/loglevel
```

To log debug messages from the Sun Cluster Data services and the most common debugging information from the Messaging Server Agents, edit the `syslog.conf` file. For example, to log all `syslog` messages to the file `/var/adm/clusterlog`, add the following line to the `syslog.conf` file:

```
daemon.debug /var/adm/clusterlog
```

This will log all messages at the following levels (`emerg`, `alert`, `critical`, `error`, `warning`, `notice`, `information`, `debug`). See `syslog.conf` man page for more information

Now restart the `syslogd` daemon:

```
pkill -HUP syslogd
```

3.4.4

Binding IP Addresses on a Server

If you are using the Symmetric or N + 1 high availability models, there are some additional things you should be aware of during configuration in order to prepare the Sun Cluster Server for Messaging Server.

Messaging Server running on a server requires that the correct IP address binds it. This is required for proper configuration of Messaging in an HA environment.

Part of configuring Messaging Server for HA involves configuring the interface address on which the Messaging Servers bind and listen for connections. By default, the servers bind to all available interface addresses. However, in an HA environment, you want the servers to bind specifically to the interface address associated with an HA logical host name.

A script is therefore provided to configure the interface address used by the servers belonging to a given Messaging Server instance. Note that the script identifies the interface address by means of the IP address which you have or will be associating with the HA logical host name used by the servers.

The script effects the configuration changes by modifying or creating the following configuration files. For the file

msg-svr-base/config/dispatcher.cnf

it adds or changes `INTERFACE_ADDRESS` option for the SMTP and SMTP Submit servers. For the file

msg-svr-base/config/job_controller.cnf

it adds or changes the `INTERFACE_ADDRESS` option for the Job Controller.

Finally it sets the `configutil service.listenaddr` and `service.http.smtphost` parameters used by the POP, IMAP, and Messenger Express HTTP servers.

Note that the original configuration files, if any, are renamed to `*.pre-ha`.

Run the script as follows:

▼ To Bind IP Addresses on a Server

- 1 Become superuser.
- 2 Execute *msg-svr-base/sbin/ha_ip_config*
- 3 The script presents the questions described below. The script may be aborted by typing `control-d` in response to any of the questions. Default answers to the questions will appear within square brackets, []. To accept the default answer, simply press the RETURN key.
 - a. Logical IP address: Specify the IP address assigned to the logical host name which Messaging Server will be using. The IP address must be specified in dotted decimal form, for example, 123.456.78.90.

The logical IP address is automatically set in the `configutil` parameter `service.http.smtphost` which allows you to see which machine is running your messaging system in a cluster. For example, if you are using Messenger Express, your server will be able to determine from which mail host to send outgoing mail.
 - b. Messaging Server Base (*msg-svr-base*): Specify the absolute path to the top-level directory in which Messaging Server is installed.
 - c. Do you wish to change any of the above choices: answer "no" to accept your answers and effect the configuration change. Answer "yes" if you wish to alter your answers.

Note – In addition, the `ha_ip_config` script automatically enables two new processes `watcher` and `msprobe` with the following parameters: `local.autorestart` and `local.watcher.enable`. These new parameters help to monitor the health of the messaging server. Process failures and unresponsive services result in log messages indicating specific failures. The cluster agents now monitor the `watcher` process and failover whenever it exits. Note that the parameters must be enabled in order for Sun Cluster to function properly.

For more information on the `watcher` and `msprobe` processes, see “[4.5 Automatic Restart of Failed or Unresponsive Services](#)” on page 127

3.4.5

Helpful Sun Cluster Commands to Manage Messaging HA

To enable the Messaging Server Resource:

```
# scswitch -e -j messaging-resource
```

To disable the Messaging Server Resource:

```
# scswitch -n -j cal-resource
```

To list all the resources and the resource groups:

```
# scstat -pvv
```

To determine the process monitoring facility (PMF) tag, that is, the process monitored by PMF:

```
# pmfadm -L
```

To list all the resources and resource groups and their status:

```
# scstat -g
```

To manage the Sun Cluster:

```
scsetup
```

3.5 Veritas Cluster Server Agent Installation

Messaging Server can be configured with Veritas Cluster Server 3.5, 4.0, 4.1, and 5.0.

Be sure to review the Veritas Cluster Server documentation prior to following these procedures.

After installing Messaging Server using the Communications Suite Installer and configuring HA, be sure to review “[3.4.4 Binding IP Addresses on a Server](#)” on page 114 for additional steps associated with configuring HA support. This section contains the following subsections:

- [“3.5.1 Veritas Cluster Server Requirements” on page 116](#)
- [“3.5.2 VCS Installation and Configuration Notes” on page 116](#)
- [“3.5.3 MsgSrv Attributes and Arguments” on page 119](#)

3.5.1

Veritas Cluster Server Requirements

- Veritas Cluster Software is already installed and configured as described in the following instructions (“[3.5.2 VCS Installation and Configuration Notes](#)” on page 116) along with the Messaging Server software on both nodes.

3.5.2

VCS Installation and Configuration Notes

The following instructions describe how to configure Messaging Server as an HA service, by using Veritas Cluster Server.

The default `main.cf` configuration file sets up a resource group called `ClusterService` that launches the `VCSweb` application. This group includes network logical host IP resources like `csgnic` and `webip`. In addition, the `ntfr` resource is created for event notification.

▼ **To Configure Messaging Server as an HA Service by Using Veritas Cluster Server**

1 Launch Cluster Explorer from one of the nodes.

Note that these Veritas Cluster Server instructions assume you are using the graphical user interface to configure Messaging Server as an HA service.

To launch Cluster Explorer, run the following command:

```
# /opt/VRTSvcs/bin/hagui
```

The VRTScscm package must be installed in order to use the GUI.

2 Using the cluster explorer, add a service group called MAIL-RG.

3 Add s1ms_dg disk group resource of type DiskGroup to the service group MAIL-RG and enable it.

4 Add s1ms_mt mount resource of type Mount to the service group MAIL-RG.

a. Be sure to click the Link button to enable linking resources, if they are not already enabled.

5 Create a link between s1ms_mt and s1ms_dg. Enable the resource s1ms_mt.

The figure depicts the dependency tree:

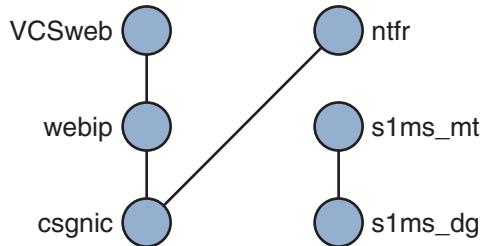


FIGURE 3-5 Veritas Cluster Server Dependency Tree 1

6 Run the Communications Suite Installer to install the Messaging Server.

a. Run the Messaging Server Initial Runtime Configuration from the primary node (for example, Node_A) to install Messaging Server.

b. Install the Veritas Cluster Server agent package, SUNWmsgvc, (located in the Messaging Server Product subdirectory on the Sun Java Communications Suite CD) by using the pkgadd(1M) command.

Messaging Server and the Veritas agent are now installed on Node_A.

- 7 Switch to the backup node (for example, Node_B).
- 8 Run the Communications Suite Installer to install Messaging Server on the backup node (Node_B).
- 9 After installing Messaging Server, use the `useconfig` utility to obviate the need for creating an additional initial runtime configuration on the backup node (Node_B). The `useconfig` utility allows you to share a single configuration between multiple nodes in an HA environment. This utility is not meant to upgrade or update an existing configuration. See “[3.3.3 Using the useconfig Utility](#)” on page 95.

The Veritas agent is now installed on Node_B.

- 10 From the Veritas Cluster Server Cluster Manager, Select Import Types... from the File menu which will display a file selection box.
- 11 Import the `MsgSrvTypes.cf` file from the `/etc/VRTSvcs/conf/config` directory. Import this type file. Note that you need to be on a cluster node to find this file.
- 12 Now create a resource of type `MsgSrv` (for example, `Mail`). This resource requires the logical host name property to be set.
- 13 The `Mail` resource depends on `s1ms_mt` and `webip`. Create links between the resources as shown in the following dependency tree:

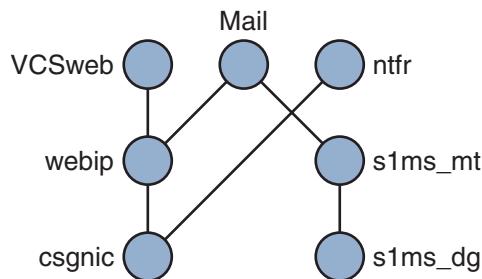


FIGURE 3–6 Veritas Cluster Dependency Tree

- a. Enable all resources and bring `Mail` online.
 - b. All servers should be started.
- 14 Switch over to Node_A and check if the High Availability configuration is working.

3.5.3

MsgSrv Attributes and Arguments

This section describes `MsgSrv` additional attributes and arguments that govern the behavior of the `mail` resource. To configure Messaging Server with Veritas Cluster Server, see [Table 3–3](#).

TABLE 3–3 Veritas Cluster Server Attributes

Attribute	Description
<code>FaultOnMonitorTimeouts</code>	If unset (=0), monitor (probe) time outs are not treated as resource fault. Recommend setting this to 2. If the monitor times out twice, the resource will be restarted or failed over.
<code>ConfInterval</code>	Time interval over which faults/restarts are counted. Previous history is erased if the service remains online for this duration. Suggest 600 seconds.
<code>ToleranceLimit</code>	Number of times the monitor should return OFFLINE for declaring the resource FAULTED. Recommend leaving this value at "0" (default).

TABLE 3–4 MsgSrv Arguments

Parameter	Description
<code>State</code>	Indicates if the service is online or not in this system. This value is not changeable by the user.
<code>LogHostName</code>	The logical host name that is associated with this instance.
<code>PrtStatus</code>	If set to TRUE, the online status is printed to the Veritas Cluster Server log file.
<code>DebugMode</code>	If set to TRUE, the debugging information is sent to the Veritas Cluster Server log file.

3.6 Unconfiguring High Availability

This section describes how to unconfigure high availability. To uninstall high availability, follow the instructions in your Veritas or Sun Cluster documentation.

The High Availability unconfiguration instructions differ depending on whether you are removing Veritas Cluster Server or Sun Cluster.

The following topics are covered in this section:

- “[To Unconfigure the Veritas Cluster Server](#)” on page 120

▼ To Unconfigure the Veritas Cluster Server

This section describes how to unconfigure the high availability components for Veritas Cluster Server:

- 1 Bring the MAIL-RG service group offline and disable its resources.**
- 2 Remove the dependencies between the mail resource, the logical_IP resource, and the mountshared resource.**
- 3 Bring the MAIL-RG service group back online so the sharedg resource is available.**
- 4 Delete all of the Veritas Cluster Server resources created during installation.**
- 5 Stop the Veritas Cluster Server and remove following files on both nodes:**
`/etc/VRTSvcs/conf/config/MsgSrvTypes.cf
/opt/VRTSvcs/bin/MsgSrv/online
/opt/VRTSvcs/bin/MsgSrv/offline
/opt/VRTSvcs/bin/MsgSrv/clean
/opt/VRTSvcs/bin/MsgSrv/monitor
/opt/VRTSvcs/bin/MsgSrv/sub.pl`
- 6 Remove the Messaging Server entries from the /etc/VRTSvcs/conf/config/main.cf file on both nodes.**
- 7 Remove the /opt/VRTSvcs/bin/MsgSrv/ directory from both nodes.**

Configuring General Messaging Capabilities

This chapter describes the general Messaging Server tasks—such as starting and stopping services and configuring directory access by using command-line utilities. Tasks specific to individual Messaging Server services—such as POP, IMAP, HTTP, and SMTP—are described in subsequent chapters. This chapter contains the following sections:

- “[4.1 To Modify Your Passwords](#)” on page 121
- “[4.2 Managing Mail Users, Mailing Lists and Domains](#)” on page 122
- “[4.3 Managing Messaging Server with Sun ONE Console](#)” on page 124
- “[4.4 Starting and Stopping Services](#)” on page 124
- “[4.5 Automatic Restart of Failed or Unresponsive Services](#)” on page 127
- “[4.6 To Schedule Automatic Tasks](#)” on page 129
- “[4.7 To Configure a Greeting Message](#)” on page 131
- “[4.8 To Set a User-Preferred Language](#)” on page 133
- “[4.9 To Customize Directory Lookups](#)” on page 134
- “[4.10 Encryption Settings](#)” on page 135
- “[4.11 Setting a Failover LDAP Server](#)” on page 136
- “[4.12 Email Security Concerns](#)” on page 136

4.1 To Modify Your Passwords

Because you set up a number of administrators with the same password during initial configuration (see “[1.3 Creating the Initial Messaging Server Runtime Configuration](#)” on page 65), you might want to change the passwords of those administrators.

Refer to [Table 4–1](#), which shows the parameters where default passwords are set up during initial runtime configuration and the utilities you can use to change them. For those parameters that use the `configutil` utility, see “[configutil](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for complete syntax and usage.

TABLE 4-1 Passwords Set in Messaging Server Initial Runtime Configuration

Parameter	Description
<code>local.ugldapbindcred</code>	Password for the user/group administrator set through the <code>configutil</code> utility.
<code>local.service.pab.ldappasswd</code>	Password for user specified by Bind DN for PAB searches set through the <code>configutil</code> utility.
SSL passwords for key files	Passwords that are directly set in the <code>sslpwpassword.conf</code> file.
Service Administrator Credentials	These are credentials that are directly set in your LDAP Directory (with the <code>ldapmodify</code> command).
Service Administrator for Delegated Administrator	You will only need to change the password of this administrator if you have enabled Sun LDAP Schema 1 and you are using the iPlanet Delegated Administrator utility.
	To change the password of the Delegated Administrator Service Administrator, you can do so by modifying your LDAP Directory (with the <code>ldapmodify</code> command), or the Delegated Administrator UI.
Store Administrator	To change the password of the Store Administrator, you can do so by modifying your LDAP Directory (with the <code>ldapmodify</code> command).

The following example uses the `local.enduseradmincred` `configutil` parameter to change the password of the end user administrator.

```
configutil -o local.enduseradmincred -v newpassword
```

4.2 Managing Mail Users, Mailing Lists and Domains

All user, mailing list and domain information is stored as entries in an LDAP directory. An LDAP directory can contain a wide range of information about an organization's employees, members, clients, or other types of individuals that in one way or another "belong" to the organization. These individuals constitute the *users* of the organization.

In the LDAP directory, the information about users is structured for efficient searching, with each user entry identified by a set of attributes. Directory attributes associated with a user can include the user's name and other identification, division membership, job classification, physical location, name of manager, names of direct reports, access permission to various parts of the organization, and preferences of various kinds.

In an organization with electronic messaging services, many if not all users hold mail accounts. For Messaging Server, mail-account information is not stored locally on the server; it is part of the LDAP user directory. The information for each mail account is stored as mail attributes attached to a user's entry in the directory.

Creating and managing mail users and mailing lists consists of creating and modifying user and mailing list entries in the directory. This is done using the Delegated Administrator for Sun LDAP Schema 2 and the iPlanet Delegated Administrator for Messaging (for Sun LDAP Schema 1), Delegated Administrator command line utilities, or by directly modifying the LDAP directory for Sun LDAP Schema 1.

▼ To Remove a User from Messaging Server

- 1 **Mark the user as deleted by running the `commadmin user delete` command.** (See the [Chapter 5, “Command Line Utilities,” in Sun Java System Delegated Administrator 6.4 Administration Guide](#).)
- 2 **Remove services from the user.**
A service can be a mailbox or a calendar. For the current version of Messaging Server, the program is called `msuserpurge`. (See “[msuserpurge](#)” in [Sun Java System Messaging Server 6.3 Administration Reference](#)). For calendar services, the program is `csclean`. (See the [Sun Java System Calendar Server 6.3 Administration Guide](#).)
- 3 **Permanently remove the user, by invoking the `commadmin domain purge` command.**

▼ To Remove a Domain from Messaging Server

- 1 **Mark the domain as deleted by running the `commadmin domain delete` command.** (See the [Chapter 5, “Command Line Utilities,” in Sun Java System Delegated Administrator 6.4 Administration Guide](#).)
- 2 **Remove services from the users of that domain.**
A service can be a mailbox or a calendar. For Messaging Server, the program is called `msuserpurge`. (See “[msuserpurge](#)” in [Sun Java System Messaging Server 6.3 Administration Reference](#)). For calendar services, the program is `csclean`. (See [Sun Java System Calendar Server Administration Guide](#).)
- 3 **Permanently remove the domain, by invoking the `commadmin domain purge` command.**

4.3 Managing Messaging Server with Sun ONE Console

The Sun ONE Administration Console is no longer supported in Messaging Server. Use the equivalent command line interfaces.

4.4 Starting and Stopping Services

Services are started and stopped differently depending on whether they are installed in an HA environment or not.

4.4.1 To Start and Stop Services in an HA Environment

While Messaging Server is running under HA control, you cannot use the normal Messaging Server start, restart, and stop commands to control individual Messaging Server services. If you attempt a `stop-msg` in an HA deployment, the system warns that it has detected an HA setup and will tell you how to properly stop the system.

The appropriate start, stop and restart commands are shown in the tables below. Note that there are no specific HA commands to individually start, restart, or stop other Messaging Server services (for example, SMTP). However, you can run a `stop-msg service` command to stop/restart individual servers such as `imap`, `pop` or `sched`.

Sun Cluster's finest granularity is that of an individual resource. Since Messaging Server is known to Sun Cluster as a resource, `scswitch` commands affect all Messaging Server services as a whole.

TABLE 4-2 Start, Stop, Restart in a Sun Cluster 3.0/3.1 Environment

Action	Individual Resource	Entire Resource Group
Start	<code>scswitch -e -j resource</code>	<code>sscswitch -Z -g resource_group</code>
Restart	<code>scswitch -n -j resource</code> <code>scswitch -e -j resource</code>	<code>scswitch -R -g resource_group</code>
Stop	<code>scswitch -n -j resource</code>	<code>scswitch -F -g resource_group</code>

TABLE 4-3 Start, Stop, Restart in Veritas 3.5, 4.0, 4.1 and 5.0 Environments

Action	Individual Resource	Entire Resource Group
Start	<code>hares -online resource -sys system</code>	<code>hagrp -online group -sys system</code>

TABLE 4-3 Start, Stop, Restart in Veritas 3.5, 4.0, 4.1 and 5.0 Environments *(Continued)*

Action	Individual Resource	Entire Resource Group
Restart	<code>hares -offline resource -sys system</code> <code>hares -online resource -sys system</code>	<code>hagrp -offline group -sys system</code> <code>hagrp -online group -sys system</code>
Stop	<code>hares -offline resource -sys system</code>	<code>hagrp -offline group -sys system</code>

4.4.2

To Start and Stop Services in a non-HA Environment

Start and stop services from the command line using the commands

`msg-svr-base/sbin/start-msg` and `msg-svr-base/sbin/stop-msg`. While it is possible to start and stop services individually using the command template: `msg-svr-base/sbin/stop-msg service` (where service can be `smtp`, `imap`, `pop`, `store`, `http`, `ens`, or `sched`) it is not recommended except in specific tasks as described in this manual. Certain services have dependencies on other services and must be started in a prescribed order. Complications can arise when trying to start services on their own. For this reason, you should start and stop all the services together using the `start-msg` and `stop-msg` commands.

Note – You must first enable services such as POP, IMAP, and HTTP, before starting or stopping them. For more information, see “[5.1.1 Enabling and Disabling Services](#)” on page 138.

Important: If a server process crashes, other processes may hang as they wait for locks held by the server process that crashed. If you are not using automatic restart (see “[4.5 Automatic Restart of Failed or Unresponsive Services](#)” on page 127), and if any server process crashes, you should stop all processes, then restart all processes. This includes the POP, IMAP, HTTP, and MTA processes, as well as the `stored` (message store) process, and any utilities that modify the message store, such as `mboxutil`, `deliver`, `reconstruct`, `readership`, or `upgrade`.

▼ To Start Up, Shut Down, or View the Status of Any Messaging Services

Again, it is not recommended to shut down individual services except in the specific tasks as described in various parts of this manual. Certain services have dependencies on other services and must be started in a prescribed order. Complications can arise when trying to start services on their own. For this reason, you should start and stop all the services together using the `start-msg` and `stop-msg` commands.

- Use the `start-msg` and `stop-msg` commands to start or stop any of the messaging services.
Examples:

`msg-svr-base/sbin/start-msg imap`

`msg-svr-base/sbin/stop-msg pop`

`msg-svr-base/sbin/stop-msg sched`

msg-svr-base/sbin/stop-msg smtp

The services must be enabled in order to stop or start them. See “[4.4.2.1 To Specify What Services Can Be Started](#)” on page 126.

Note – The `start-msg` and `stop-msg` command start and stop all of the MTA services, not just the SMTP server. If you want more granular control when starting or stopping the MTA services, you can use the `start/stop-msg` command for the dispatcher and the job controller. For more information, see “[start-msg](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* and “[stop-msg](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

4.4.2.1

To Specify What Services Can Be Started

By default the following services are started with `start-msg`:

```
#!/bin/sh
#./start-msg
Connecting to watcher ...
Launching watcher ...
Starting ens server .... 21132
Starting store server .... 21133
checking store server status ... ready
Starting imap server .... 21135
Starting pop server .... 21138
Starting http server .... 21141
Starting sched server .... 21143
Starting dispatcher server .... 21144
Starting job_controller server .... 21146
```

These can be controlled by enabling or disabling the `configutil` parameters: `service imap enable`, `service pop enable`, `service http enable`, `local smsgateway enable`, `local snmp enable`, `local imta enable`, `local mmp enable`, `local ens enable`, and `local sched enable`. Note that you need to set both `service imap enable` and `service imap enablelesslport to 0` in order to disable IMAP. The same goes for POP and HTTP. See the “[configutil Parameters](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for details on how these work.

4.4.3

Starting and Stopping a Messaging Server Running in MTA-only Mode

To start an MTA-only system, you should also start `imsched`. Before you do this, remove any scheduled jobs that are not appropriate to your installations.

`imsched` is an individual component of Messaging Server which must be started separately if you are not starting all of Messaging Server. If you start your MTA-only system using `start-msg imtaor start-msg smtp`, then you will not be running the `imsched` process.

To run messaging server in MTA mode only (no store/imap/pop/http processes), then you can either select the MTA to be only installed/configured during the configuration of messaging server after initial install (*msg_base/sbin/configure*) or manually disable the message store and *mshttp* process using the following *configutil* commands:

```
./configutil -o local.store.enable -v 0
./configutil -o service.http.enable -v 0
```

Once you have disabled http and other store processes, you can then start messaging server by running the following:

```
# ./start-msg
bash-3.00# ./start-msg
Connecting to watcher ...
Launching watcher ... 4034
Starting ens server ... 4035
Starting sched server ... 4036
Starting dispatcher server .... 4038
Starting job_controller server .... 4042
```

Not that all the appropriate processes are started including *imsched* and *imta*. This way the customer doesn't have to remember to start the *sched* process.

4.5 Automatic Restart of Failed or Unresponsive Services

Messaging Server provides two processes called *watcher* and *msprobe* that transparently monitor services and automatically restart them if they crash or become unresponsive (the services hangs). *watcher* monitors server crashes. *msprobe* monitors server hangs by checking the response time. When a server fails or stops responding to requests, it is automatically restarted. [Table 4–4](#) shows the services monitored by each utility.

TABLE 4–4 Services Monitored by *watcher* and *msprobe*

<i>watcher</i> (crash)	<i>msprobe</i> (unresponsive hang)
IMAP, POP, HTTP, job controller, dispatcher, message store (<i>stored</i>), <i>imsched</i> , MMP, (LMTP/SMTP servers are monitored by the dispatcher and LMTP/SMTP clients are monitored by the <i>job_controller</i> .)	IMAP, POP, HTTP, cert, job controller, message store (<i>stored</i>), <i>imsched</i> , ENS, LMTP, SMTP

Setting *local.watcher.enable=on* (default) will monitor process failures and unresponsive services and will log error messages to the default log file indicating specific failures. To enable automatic server restart, set the *configutil* parameter *local.autorestart* to yes. By default, this parameter is set to no.

If any of the message store services fail or freeze, all message store services that were enabled at start-up are restarted. For example, if `imapd` fails, at least, `stored` and `imapd` are restarted. If other message store services were running, such as the POP or HTTP servers, then those will be restarted as well, whether or not they failed.

Automatic restart also works if a message store utility fails or freezes. For example, if `mboxutil` fails or freezes, the system will automatically restart all the message store servers. Note, however, that it will not restart the utility. `msprobe` runs every 10 minutes. Service and process restarts will be performed up to two times within a 10 minute period (configurable using `local.autorestart.timeout`).

Whether or not `local.autorestart` is set to yes, the system still monitors the services and sends failure or non-response error messages to the console and `msg-svr-base/data/log/watcher` listens to port 49994 by default, but this is configurable with `local.watcher.port`.

A watcher log file is generated in `msg-svr-base/data/log/watcher`. This log file is not managed by the logging system (no rollover or purging) and records all server starts and stops. An example log is shown below:

```
watcher process 13425 started at Tue Oct 21 15:29:44 2003

Watched 'imapd' process 13428 exited abnormally
Received request to restart: store imap pop http
Connecting to watcher ...
Stopping http server 13440 .... done
Stopping pop server 13431 ... done
Stopping pop server 13434 ... done
Stopping pop server 13435 ... done
Stopping pop server 13433 ... done
imap server is not running
Stopping store server 13426 .... done
Starting store server .... 13457
checking store server status ..... ready
Starting imap server ..... 13459
Starting pop server ..... 13462
Starting http server ..... 13471
```

See “[27.8.9 Monitoring Using msprobe and watcher Functions](#)” on page 886 for more details on how to configure this feature.

`msprobe` is controlled by `imsched`. If `imsched` crashes, this event will be detected by `watcher` and trigger a restart (if autorestart is enabled). However, in the rare occurrence of `imsched` hanging, you will need to kill `imsched` with a kill `imsched_pid`, which will cause the `watcher` to restart it.

4.5.1 Automatic Restart in High Availability Deployments

Automatic restart in high availability deployments require the following configutil parameters to be set:

TABLE 4–5 HA Automatic Restart Parameters

Parameter	Description/HAValue
local.watcher.enable	Enable watcher on start-msg startup. Default is yes.
local.autorestart	Enable automatic restart of failed or frozen (unresponsive) servers including IMAP, POP, HTTP, job controller, dispatcher, and MMP servers. Default is No.
local.autorestart.timeout	Failure retry time-out. If a server fails more than once during this designated period of time, then the system will stop trying to restart this server. If this happens in an HA system, Messaging Server is shutdown and a failover to the other system occurs. The value (set in seconds) should be set to a period value longer than the msprobe interval. (See local.schedule.msprobe below). Default is 600.
local.schedule.msprobe	msprobe run schedule. A crontab style schedule string (see Table 20–10). Default is 5,15,25,35,45,55 * * * * lib/msprobe To disable: set local.schedule.msprobe.enable to NO.

4.6 To Schedule Automatic Tasks

Messaging Server provides a general task scheduling mechanism using a process called imsched. It is intended for scheduling Messaging Server processes. It is enabled by setting the local.schedule.taskname configutil parameter. If you modify the schedule, you must either restart the scheduler with the command stop-msg sched and start-msg sched, or you can refresh the scheduler process (refresh sched).

This parameter requires a command and a schedule on which to execute the command. The format is as follows:

`configutil -o local.schedule.taskname -v "schedule"`

taskname is a unique name for this command/schedule combination.

schedule has the format:

minute hour day-of-month month-of-year day-of-week command args

command args can be any Messaging Server command and its arguments. Paths can be relative to msg-svr-base or absolute paths. See “[4.6.2 Pre-defined Automatic Tasks](#)” on page 130 for relative path examples.

minute hour day-of-month month-of-year day-of-week is the schedule for running the command. It follows the UNIX crontab format.

The values are separated by a space or tab and can be 0-59, 0-23, 1-31, 1-12 or 0-6 (with 0=Sunday) respectively. Each time field can be either an asterisk (meaning all legal values), a list of comma-separated values, or a range of two values separated by a hyphen. Note that days can be specified by both day of the month and day of the week and both will be required if specified. For example, setting the 17th day of the month and Tuesday will only run the command on the 17th day of a month when it is Tuesday. See [Table 20–10](#)

Note that if you modify scheduler, you must either restart the scheduler with the command `stop-msg sched` and `start-msg sched`, or you can refresh the scheduler process:

```
refresh sched
```

To disable a scheduled task, run the following:

```
# configutil -o local.schedule.taskname.enable -v no  
# refresh sched
```

4.6.1 Scheduler Examples

Run `imexpire` at 12:30am, 8:30am, and 4:30pm:

```
# configutil -o local.schedule.rm_messages -v "30 0,8,16 * * * /opt/SUNWmsgsr/sbin/imexpire"
```

Display MTA channel queue message counters every 20 minutes:

```
# configutil -o local.schedule.counters -v "0,20,40 * * * * /opt/SUNWmsgsr/sbin/ims  
# imta qm counters > /tmp/temp.txt"
```

Run `imsbackup` Monday through Friday at midnight (12AM):

```
# configutil -o local.schedule.msbackup -v "0 0 * * 1-5 /opt/SUNWmsgsr/sbin/imsbackup -f \  
backupfile /primary"
```

4.6.2 Pre-defined Automatic Tasks

At installation, Messaging Server creates, schedules and enables a set of pre-defined automatic tasks. These are shown below.

The following automatic tasks are set and enabled for the message store:

```
local.schedule.expire = "0 23 * * * sbin/imexpire"  
local.schedule.expire.enable = 1  
local.schedule.snapshotverify = "0 0,4,8,12,16,20 * * * sbin/imdbverify -m"  
local.schedule.snapshotverify.enable = 1
```

The following automatic tasks are set and enabled for the MTA:

```
local.schedule.purge="0 0,4,8,12,16,20 * * * sbin/imsimta purge -num=5"
local.schedule.purge.enable = 1
local.schedule.return_job = "30 0 * * * lib/return_job"
local.schedule.return_job.enable = 1
```

The following automatic tasks are set and enabled for the message store:

```
local.schedule.msprobe = "5,15,25,35,45,55 * * * lib/msprobe"
local.schedule.msprobe.enable = 1
```

4.7 To Configure a Greeting Message

Messaging Server allows you to create an email greeting message to be sent to each new user.

▼ To Create a New User Greeting

- To create a new-user greeting use the command line:

```
configutil -o gen.newuserforms -v Message
```

Where *Message* must contain a header (with at least a subject line), followed by \$\$, then the message body. The \$ represents a new line.

For example, to enable this parameter, you can set the following configuration variables:

```
configutil -o gen.newuserforms -v 'Subject: Welcome!! $$ Sesta.com welcomes you
to the premier internet experience in Dafandzadgad!'
```

Depending on the shell that you are using, it might be necessary to append a special character before \$ to escape the special meaning of \$. (\$ is often the escape character for the shell.)

4.7.1

To Set a Per-Domain Greeting Message

Whenever you create a new hosted domain, it is a good idea to create per-domain greeting messages for your supported languages. If this is not done, the generic greeting message set by gen.newuserforms will be sent.

You can set a greeting message for new users in each domain. The message can vary depending on the user's, the domain's, or the site's preferred language. This is done by setting the mailDomainWelcomeMessage attribute in the desired LDAP domain entry. The attribute syntax is as follows:

```
mailDomainWelcomeMessage;lang=user_prefLang
mailDomainWelcomeMessage;lang=domain_prefLang
mailDomainWelcomeMessage;lang=gen.sitelanguage
```

The following example sets the domain welcome message for English:

```
mailDomainWelcomeMessage;lang-en: Subject: Welcome!! $$Welcome to the mail  
system.
```

The following example sets the domain welcome message for French:

```
mailDomainWelcomeMessage;lang-fr: Subject: Bienvenue!! $$Bienvenue a siroe.com!
```

Using the above examples, we assume the following:

- the domain is siroe.com
- a new user belongs to this domain
- the user's preferred language is French as specified by the LDAP attribute `preferredlanguage`
- siroe.com has the above English and French welcome messages available
- the site language is en as specified by `gen.sitelanguage`.

For a list of supported locales and their language value tag, see the ([Directory Server Reference Manual](#) (<http://docs.sun.com>)).

When the user logs in for the first time, they will receive the French greeting. If the French welcome message isn't available, they will get the English greeting.

4.7.1.1

Greeting Message Theory of Operations

Greeting messages can be set by both the LDAP attribute `mailDomainWelcomeMessage` and the configutil parameter `gen.newuserforms`. The order in which a message will get chosen, with the top one having the highest preference, is shown below:

```
mailDomainWelcomeMessage;lang-user_prefLang  
mailDomainWelcomeMessage;lang-domain_prefLang  
mailDomainWelcomeMessage;lang-gen.sitelanguage  
mailDomainWelcomeMessage  
gen.newuserforms;lang-"$user-prefLang"  
gen.newuserforms;lang-"$domain-prefLang"  
gen.newuserforms;lang-"$gen.sitelanguage"  
gen.newuserforms
```

The algorithm works as follows: if there are no domains (or there are, but there is no per domain welcome message provisioned for them), a welcome message is configured with the `gen.newuserforms` parameter, if specified. If a user has a preferred language (set with the `preferredlanguage` LDAP attribute) and `gen.newuserforms; lang-user_prefLang` is set, the user will receive that welcome message at the time of their first log in to the server. If `gen.newuserforms; lang-gen.sitelanguage` is set, and `preferredlanguage` is not set, but the site language is set (using `gen.sitelanguage` parameter), user will receive that message. If no

language tag parameter is set and a untagged gen.newuserforms is set, then that message will be sent to the user. If none of the values are set, user will not receive any welcome message.

If the user is in a domain, then similar to the discussion above, the user might receive one of mailDomainWelcomeMessage; lang-xx, depending on which one is available in the list and in the order given.

Example: Domain is siroe.com. The domain preferred language is German (de). But the new user in this domain has preferred language of Turkish (tr). Site language is English. The following values are available (mailDomainWelcomeMessage are attributes of the domain siroe.com):

```
mailDomainWelcomeMessage; lang-fr  
mailDomainWelcomeMessage; lang-ja  
gen.newuserforms; lang-de  
gen.newuserforms; lang-en  
gen.newuserforms
```

According to the algorithm, the message sent to user will be gen.newuserforms; lang-de.

4.8 To Set a User-Preferred Language

Administrators can set a preferred language for the GUI and server-generated messages by setting the attribute preferredLanguage in the user's LDAP entry.

When the server sends messages to users outside of the server's administrative domain it does not know what their preferred language is unless it is responding to an incoming message with a preferred language specified in the incoming message's header. The header fields (Accept-Language, Preferred-Language or X-Accept-Language) are set according to attributes specified in the user's mail client.

If there are multiple settings for the preferred language—for example, if a user has a preferred language attribute stored in the Directory Server and also has a preferred language specified in their mail client—the server chooses the preferred language in the following order:

1. The Accept-Language header field of the original message.
2. The Preferred-Language header field of the original message.
3. The X-Accept-Language header field of the original message.
4. The preferred language attribute of the sender (if found in the LDAP directory).

4.8.1

To Set a Domain Preferred Language

A domain preferred language is a default language specified for a particular domain. For example, you may wish to specify Spanish for a domain called `mexico.siroe.com`.

Administrators can set a domain preferred language by setting the attribute `preferredLanguage` in the domain's LDAP entry.

▼ To Specify a Site Language

You can specify a default site language for your server as follows. The site language will be used to send language-specific versions of messages if no user preferred language is set.

- **Command Line: Specify a site language as follows:**

```
configutil -o gen.sitelanguage -v value
```

where `value` is one of the local supported languages. See Chapter 5 of *Sun Java System Directory Server 5 2005Q1 Administration Guide* for a list of supported locales and the language value tag.

4.9 To Customize Directory Lookups

Messaging Server cannot function without an LDAP-based directory system such as the Sun Java System Directory Server. Messaging Server requires directory access for a number of purposes. For example:

- When you create or update account information for mail users or mail groups, the information is stored in a directory called the *user directory*.
- When routing messages and delivering mail to mailboxes, Messaging Server looks up information about the sender or recipients in the user directory.
- When authenticating a user for mail routing lookups.

Reconfiguring your Messaging Server to connect to a different user directory for user and group lookups is strictly optional. In most cases, the user directory that defines your server's administrative domain is the one used by all servers in the domain.

▼ To Modify the Messaging Server LDAP User-lookup Settings

- The commands for the user-directory connection settings are shown below, but first set the LDAP and PAB password as follows:

- Modify the password for the user specified in the configuration attribute `local.ugldapbinddn`. This user account exists in the directory server specified in configuration attribute `local.ugldaphost`.
- If the same account is used for PAB access, specified in the attributes `local.service.pab.ldapbinddn` and `local.service.pab.ldaphost`, then the password stored in `local.service.pab.ldappasswd` must be updated.

To specify whether to use Messaging Server specific directory settings:

```
configutil -o local.ugldapuselocal -v [ yes | no ]
```

Host name is the name of the host machine on which the directory containing your installation’s user information resides. This is typically not the same as the Messaging Server host, although for very small installations it might be. To specify the LDAP host name for user lookup:

```
configutil -o local.ugldaphost -v name[:port_number]
```

Port number is the port number on the directory host that Messaging Server must use to access the directory for user lookup. This number is defined by the directory administrator, and may not necessarily be the default port number (389). To specify the LDAP port number for user lookup:

```
configutil -o local.ugldapport -v number
```

The **Base DN**: is the search base—the distinguished name of a directory entry that represents the starting point for user lookups. To speed the lookup process, the search base should be as close as possible in the directory tree to the information being sought. If your installation’s directory tree has a “people” or “users” branch, that is a reasonable starting point. To specify the LDAP base DN for user lookup:

```
configutil -o local.ugldapbasedn -v basedn
```

Bind DN: is the distinguished name that your Messaging Server uses to represent itself when it connects to the directory server for lookups. The bind DN must be the distinguished name of an entry in the user directory itself that has been given search privileges to the user portion of the directory. If the directory allows anonymous search access, you can leave this entry blank. To specify the LDAP bind DN for user lookup:

```
configutil -o local.ugldapbinddn -v binddn
```

4.10 Encryption Settings

This is described in “[23.5.2 To Enable SSL and Selecting Ciphers](#)” on page 731, which also contains background information on all security and access-control topics for Messaging Server.

4.11 Setting a Failover LDAP Server

It is possible to specify more than one LDAP server for the user/group directory so that if one fails another takes over:

▼ To Set a Failover LDAP Server

- 1 **Set local.ugldaphost to the multiple LDAP machines. Example:**

```
configutil -o local.ugldaphost -v "server1 server2 ..."
```

- 2 **Set local.ugldapuselocal to yes. This specifies that the user/group LDAP configuration data will be stored in the local configuration file. Otherwise, it is stored in LDAP. Example:**

```
configutil -o local.ugldapuselocal -v yes
```

If the

first server on the list fails, the existing LDAP connections will get recognized as down and new connections will be made. When a new LDAP connection is needed, the LDAP library will try all the LDAP servers in the order they're listed.

4.12 Email Security Concerns

By default, Messaging Server enables four private commands called XADR, which returns information about how an address is routed internally by the MTA as well as general channel information, XCIR, which returns MTA circuit check information, XGEN, which returns status information about whether a compiled configuration and compiled character set are in use, and XSTA, which returns status information about the number of messages processed and currently in the MTA channel queues. Releasing such information may constitute a breach of security for some sites.

Sites may wish to disable these commands for the tcp_local channel by adding the following lines to the file `tcp_local_options`, creating it if necessary.

```
DISABLE_ADDRESS=1  
DISABLE_CIRCUIT=1  
DISABLE_STATUS=1  
DISABLE_GENERAL=1
```

Configuring POP, IMAP, and HTTP Services

Messaging Server supports the Post Office Protocol 3 (POP3), the Internet Mail Access Protocol 4 (IMAP4), and the HyperText Transfer Protocol (HTTP) for client access to mailboxes. IMAP and POP are both Internet-standard mailbox protocols. Messenger Express, a web-enabled electronic mail program, lets end users access their mailboxes using a browser running on an Internet-connected computer system using HTTP.

This chapter describes how to configure your server to support one or more of these services by using command-line utilities.

For information on configuring Simple Mail Transfer Protocol (SMTP) services, see [Chapter 10, “About MTA Services and Configuration.”](#)

This chapter contains the following sections:

- “[5.1 General Configuration](#)” on page 137
- “[5.2 Login Requirements](#)” on page 140
- “[5.3 Performance Parameters](#)” on page 142
- “[5.4 Client Access Controls](#)” on page 145
- “[5.5 To Configure POP Services](#)” on page 145
- “[5.6 To Configure IMAP Services](#)” on page 146
- “[5.7 To Configure HTTP Services](#)” on page 151

5.1 General Configuration

Configuring the general features of the Messaging Server POP, IMAP, and HTTP services includes enabling or disabling the services, assigning port numbers, and optionally modifying service banners sent to connecting clients. This section provides background information; for the steps you follow to make these settings, see “[5.5 To Configure POP Services](#)” on page 145, “[5.6 To Configure IMAP Services](#)” on page 146 and “[5.7 To Configure HTTP Services](#)” on page 151. This section consists of the following subsections:

- “[5.1.1 Enabling and Disabling Services](#)” on page 138

- “[5.1.2 Specifying Port Numbers](#)” on page 138
- “[5.1.3 Ports for Encrypted Communications](#)” on page 139
- “[5.1.4 Service Banner](#)” on page 139

5.1.1

Enabling and Disabling Services

You can control whether any particular instance of Messaging Server makes its POP, IMAP, or HTTP service available for use. This is not the same as starting and stopping services (see “[4.4 Starting and Stopping Services](#)” on page 124); to function, POP, IMAP, or HTTP must be both enabled and started.

Enabling a service is a more “global” process than starting or stopping a service. For example, the Enable setting persists across system reboots, whereas you must restart a previously “stopped” service after a reboot.

There is no need to enable services that you do not plan to use. For example, if a Messaging Server instance is used only as a mail transfer agent (MTA), you should disable POP, IMAP, and HTTP. If it is used only for POP services, you should disable IMAP and HTTP. If it used only for web-based email, you should disable both POP and IMAP.

You can enable or disable services at the server level. This process is described in this chapter. “[4.4.2.1 To Specify What Services Can Be Started](#)” on page 126 also describes this process. You can also enable or disable services at the user level by setting the LDAP attribute `mailAllowedServiceAccess`.

5.1.2

Specifying Port Numbers

For each service, you can specify the port number that the server is to use for service connections:

- If you enable the POP service, you can specify the port number that the server is to use for POP connections. The default is 110.
- If you enable the IMAP service, you can specify the port number that the server is to use for IMAP connections. The default is 143.
- If you enable the HTTP service, you can specify the port number that the server is to use for HTTP connections. The default is 80.

You might need to specify a port number other than the default if you have, for example, two or more IMAP server instances on a single host machine, or if you are using the same host machine as both an IMAP server and a Messaging Multiplexor server. (For information about the Multiplexor, see [Chapter 7, “Configuring and Administering Multiplexor Services.”](#))

Keep the following in mind when you specify a port:

- Port numbers can be any number from 1 to 65535.
- Make sure the port you choose isn't already in use or reserved for another service.

5.1.3 Ports for Encrypted Communications

Messaging Server supports encrypted communications with IMAP, POP and HTTP clients by using the Secure Sockets Layer (SSL) protocol. For general information on support for SSL in Messaging Server, see “[23.5 Configuring Encryption and Certificate-Based Authentication](#)” on page 720.

5.1.3.1 IMAP Over SSL

You can accept the default (recommended) IMAP over SSL port number (993) or you can specify a different port for IMAP over SSL.

Messaging Server provides the option of using separate ports for IMAP and IMAP over SSL because most current IMAP clients require separate ports for them. Same-port communication with both IMAP and IMAP over SSL is an emerging standard; as long as your Messaging Server has an installed SSL certificate (see “[23.5.1 Obtaining Certificates](#)” on page 722), it can support same-port IMAP over SSL.

5.1.3.2 POP Over SSL

The default separate SSL port for POP is 995. One can also initiate SSL over normal POP port with the command “STLS” (see “[5.5 To Configure POP Services](#)” on page 145).

5.1.3.3 HTTP Over SSL

You can accept the default HTTP over SSL port number (443) or you can specify a different port for HTTPS.

5.1.4 Service Banner

When a client first connects to the Messaging Server POP or IMAP port, the server sends an identifying text string to the client. This service banner (not normally displayed to the client's user) identifies the server as Sun Java System Messaging Server, and gives the server's version number. The banner is most typically used for client debugging or problem-isolation purposes.

You can replace the default banner for the POP or IMAP service if you want a different message sent to connecting clients.

Use the configutil utility (`service imap.banner`, `service pop.banner`) to set service banners. For detailed syntax information about configutil, see the *Sun Java System Messaging Server 6.3 Administration Reference*.

5.2 Login Requirements

You can control how users are permitted to log in to the POP, IMAP, or HTTP service to retrieve mail. You can allow password-based login (for all services), and certificate-based login (for IMAP or HTTP services). This section provides background information; for the steps you follow to make these settings, see “[5.5 To Configure POP Services](#)” on page 145, “[5.6 To Configure IMAP Services](#)” on page 146 or “[5.7 To Configure HTTP Services](#)” on page 151. In addition, you can specify the valid login separator for POP logins. This section consists of the following subsections:

- “[To Set the Login Separator for POP Clients](#)” on page 140
- “[5.2.1 To Allow Log In without Using the Domain Name](#)” on page 140
- “[5.2.2 Password-Based Login](#)” on page 141
- “[5.2.3 Certificate-Based Login](#)” on page 141

▼ **To Set the Login Separator for POP Clients**

Some mail clients will not accept @ as the login separator (that is, the @ in an address like uid@domain). Examples of these clients are Netscape Messenger 4.76, Netscape Messenger 6.0, and Microsoft Outlook Express on Windows 2000. The workaround is as follows:

1 Make + a valid separator with the following command:

```
configutil -o service.loginseparator -v "@+"
```

2 Inform POP client users that they should login with + as the login separator, not @.

5.2.1

To Allow Log In without Using the Domain Name

A typical login involves the user entering a user ID followed by a separator and the domain name and then the password. Users in the default domain specified during installation, however, can log in without entering a domain name or separator.

To allow users of other domains to log in with just the user ID (that is, without having to use the domain name and separator) set `sasl.default.ldap.searchfordomain` to 0. Note that the user ID must be unique to the entire directory tree. If it is not unique, logging in without the domain name will not work.

You may wish to modify the attribute that user must enter to log in. For example if you want to allow the user to log in with a phone number (`telephoneNumber`) or employee number (`employeeID`) change the LDAP search defined by `configutil` parameter `sasl.default.ldap.searchfilter`. This parameter is a global default setting for the `inetDomainSearchFilter` per-domain attribute and follows the same syntax.

Refer to the [Sun Java System Messaging Server 6.3 Administration Reference](#) for further information on these parameters.

5.2.2

Password-Based Login

In typical messaging installations, users access their mailboxes by entering a password into their POP, IMAP or HTTP mail client. The client sends the password to the server, which uses it to authenticate the user. If the user is authenticated, the server decides, based on access-control rules, whether or not to grant the user access to certain mailboxes stored on that server.

If you allow password login, users can access POP, IMAP, or HTTP by entering a password. (Password- or SSL-based login is the only authentication method for POP services.) Passwords are stored in an LDAP directory. Directory policies determine what password policies, such as minimum length, are in effect.

If you disallow password login for IMAP or HTTP services, password-based authentication is not permitted. Users are then required to use certificate-based login, as described in the next section.

To increase the security of password transmission for IMAP and HTTP services, you can require that passwords be encrypted before they are sent to your server. You do this by selecting a minimum cipher-length requirement for login.

- If you choose 0, you do not require encryption. Passwords are sent in the clear or they are encrypted, depending on client policy.
- If you choose a nonzero value, the client must establish an SSL session with the server—using a cipher whose key length is at least the value you specify—thus encrypting any IMAP or HTTP user passwords the client sends.

If the client is configured to require encryption with key lengths greater than the maximum your server supports, or if your server is configured to require encryption with key lengths greater than what the client supports, password-based login cannot occur. For information on setting up your server to support various ciphers and key lengths, see “[23.5.2 To Enable SSL and Selecting Ciphers](#)” on page [731](#).

5.2.3

Certificate-Based Login

In addition to password-based authentication, Sun Java System servers support the authentication of users through examination of their digital certificates. Instead of presenting a password, the client presents the user’s certificate when it establishes an SSL session with the server. If the certificate is validated, the user is considered authenticated.

For instructions on setting up Messaging Server to accept certificate-based user login to the IMAP or HTTP service, see “[23.5.3 To Set Up Certificate-Based Login](#)” on page [733](#)

If you have performed the tasks required to set up certificate-based login, both password-based and certificate-based login are supported. Then, if the client establishes an SSL session and supplies a certificate, certificate-based login is used. If the client does not use SSL or does not present a client certificate, it will send a password instead.

5.3 Performance Parameters

You can set some of the basic performance parameters for the POP, IMAP, and HTTP services of Messaging Server. Based on your hardware capacity and your user base, you can adjust these parameters for maximum efficiency of service. This section provides background information; for the steps you follow to make these settings, see “[5.5 To Configure POP Services](#)” on page 145, “[5.6 To Configure IMAP Services](#)” on page 146 or “[5.7 To Configure HTTP Services](#)” on page 151. This section consists of the following subsections:

- “[5.3.1 Number of Processes](#)” on page 142
- “[5.3.2 Number of Connections per Process](#)” on page 143
- “[5.3.3 Number of Threads per Process](#)” on page 144
- “[5.3.4 Dropping Idle Connections](#)” on page 144
- “[5.3.5 Logging Out HTTP Clients](#)” on page 145

5.3.1 Number of Processes

Messaging Server can divide its work among several executing processes, which in some cases can increase efficiency. This capability is especially useful with multiprocessor server machines, in which adjusting the number of server processes can allow more efficient distribution of multiple tasks among the hardware processors.

There is a performance overhead, however, in allocating tasks among multiple processes and in switching from one process to another. The advantage of having multiple processes diminishes with each new one added. A simple rule of thumb for most configurations is to have one process per hardware processor on your server machine, up to a maximum of perhaps 4 processes. Your optimum configuration may be different; this rule of thumb is meant only as a starting point for your own analyses.

Note: On some platforms you might also want to increase the number of processes to get around certain per-process limits (such as the maximum number of file descriptors), specific to that platform, that may affect performance.

The default number of processes is 1 each for the POP, IMAP, or HTTP service.

5.3.2

Number of Connections per Process

The more simultaneous client connections your POP, IMAP, or HTTP service can maintain, the better it is for clients. If clients are denied service because no connections are available, they must then wait until another client disconnects.

On the other hand, each open connection consumes memory resources and makes demands on the I/O subsystem of your server machine, so there is a practical limit to the number of simultaneous sessions you can expect the server to support. (You might be able to increase that limit by increasing server memory or I/O capacity.)

IMAP, HTTP, and POP have different needs in this regard:

- IMAP connections are generally long-lived compared to POP and HTTP connections. When a user connects to IMAP to download messages, the connection is usually maintained until the user quits or the connection times out. In contrast, a POP or HTTP connection is usually closed as soon as the POP or HTTP request has been serviced.
- IMAP and HTTP connections are generally very efficient compared to POP connections. Each POP reconnection requires reauthentication of the user. In contrast, an IMAP connection requires only a single authentication because the connection remains open for the duration of the IMAP session (login to logout). An HTTP connection is short, but the user need not reauthenticate for each connection because multiple connections are allowed for each HTTP session (login to logout). POP connections, therefore, involve much greater performance overhead than IMAP or HTTP connections. Messaging Server, in particular, has been designed to require very low overhead by open but idle IMAP connections and by multiple HTTP connections.

Note – For more information about HTTP session security, see “[23.2 About HTTP Security](#)” on page [714](#)

Thus, at a given moment for a given user demand, Messaging Server may be able to support many more open IMAP or HTTP connections than POP connections.

The default value for IMAP is 4000; the default value for HTTP is 6000 connections per process; the default value for POP is 600. These values represent roughly equivalent demands that can be handled by a typically configured server machine. Your optimum configuration may be different; these defaults are meant only as general guidelines.

Typically, active POP connections are much more demanding on server resources and bandwidth than active IMAP connections since IMAP connections are idle most of the time while POP connections are constantly downloading messages. Having a lower number of sessions for POP is correct. Conversely, POP connections only last as long as it takes to download email, so an active POP user is only connected a small percentage of the time, while IMAP connections stay connected between successive mail checks.

5.3.3

Number of Threads per Process

Besides supporting multiple processes, Messaging Server further improves performance by subdividing its work among multiple threads. The server's use of threads greatly increases execution efficiency, because commands in progress are not holding up the execution of other commands. Threads are created and destroyed, as needed during execution, up to the maximum number you have set.

Having more simultaneously executing threads means that more client requests can be handled without delay, so that a greater number of clients can be serviced quickly. However, there is a performance overhead to dispatching among threads, so there is a practical limit to the number of threads the server can make use of.

For POP, IMAP, and HTTP, the default maximum value is 250 threads per process. The numbers are equal despite the fact that the default number of connections for IMAP and HTTP is greater than for POP. It is assumed that the more numerous IMAP and HTTP connections can be handled efficiently with the same maximum number of threads as the fewer, but busier, POP connections. Your optimum configuration may be different, but these defaults are high enough that it is unlikely you would ever need to increase them; the defaults should provide reasonable performance for most installations.

5.3.4

Dropping Idle Connections

To reclaim system resources used by connections from unresponsive clients, the IMAP4, POP3, and HTTP protocols permit the server to unilaterally drop connections that have been idle for a certain amount of time.

The respective protocol specifications require the server to keep an idle connection open for a minimum amount of time. The default times are 10 minutes for POP, 30 minutes for IMAP, 3 minutes for HTTP. You can increase the idle times beyond the default values, but you cannot make them less.

If a POP or IMAP connection is dropped, the user must reauthenticate to establish a new connection. In contrast, if an HTTP connection is dropped, the user need not reauthenticate because the HTTP session remains open. For more information about HTTP session security, see “[23.2 About HTTP Security](#)” on page 714.

Idle POP connections are usually caused by some problem (such as a crash or hang) that makes the client unresponsive. Idle IMAP connections, on the other hand, are a normal occurrence. To keep IMAP users from being disconnected unilaterally, IMAP clients typically send a command to the IMAP server at some regular interval that is less than 30 minutes.

5.3.5 Logging Out HTTP Clients

An HTTP session can persist across multiple connections. HTTP clients are not logged out when a connection is dropped. However, if an HTTP session remains idle for a specified time period, the server will automatically drop the HTTP session and the client is logged out (the default time period is 2 hours). When the session is dropped, the client's session ID becomes invalid and the client must reauthenticate to establish another session. For more information about HTTP security and session ID's, see “[23.2 About HTTP Security](#)” on page [714](#).

5.4 Client Access Controls

Messaging Server includes access-control features that allow you to determine which clients can gain access to its POP, IMAP, or HTTP messaging services (and SMTP as well). You can create flexible access filters that allow or deny access to clients based on a variety of criteria.

Client access control is an important security feature of Messaging Server. For information on creating client access-control filters and examples of their use, see “[23.7 Configuring Client Access to POP, IMAP, and HTTP Services](#)” on page [737](#) and “[23.9 Configuring Client Access to SMTP Services](#)” on page [749](#).

5.5 To Configure POP Services

You can perform basic configuration of the Messaging Server POP service by using the `configutil` command. Some of the more common POP services options are given in this section. A complete listing can be found in the “[configutil Parameters](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

Note – For the POP service, password-based login is automatically enabled.

For more information, see also:

- “[5.1.1 Enabling and Disabling Services](#)” on page [138](#)
- “[To Set the Login Separator for POP Clients](#)” on page [140](#)
- “[5.1.2 Specifying Port Numbers](#)” on page [138](#)
- “[5.3.2 Number of Connections per Process](#)” on page [143](#)
- “[5.3.4 Dropping Idle Connections](#)” on page [144](#)
- “[5.3.3 Number of Threads per Process](#)” on page [144](#)
- “[5.3.1 Number of Processes](#)” on page [142](#)

To enable or disable the POP service:

```
configutil -o service.pop.enable -v [ yes | no ]
```

To specify the port number:

```
configutil -o service.pop.port -v number
```

To set the maximum number of network connections per process (see “[5.3.2 Number of Connections per Process](#)” on page 143 for details):

```
configutil -o service.pop.maxsessions -v number
```

To set the maximum idle time for connections (see “[5.3.4 Dropping Idle Connections](#)” on page 144 for details):

```
configutil -o service.pop.idletimeout -v number
```

To set the maximum number of threads per process (see “[5.3.3 Number of Threads per Process](#)” on page 144 for more information):

```
configutil -o service.pop.maxthreads -v number
```

To set the maximum number of processes (see “[5.3.1 Number of Processes](#)” on page 142 for additional information):

```
configutil -o service.pop.numprocesses -v number
```

To enable POP over SSL:

```
configutil -o service.pop.enablesslport -v 1  
configutil -o service.pop.sslport -v 995
```

TLS is also supported if SSL is configured correctly.

To specify a protocol welcome banner:

```
configutil -o service.pop.banner -v banner
```

5.6 To Configure IMAP Services

You can perform basic configuration of the Messaging Server IMAP service by using the `configutil` command. Some of the more common IMAP services options are given in this section. A complete listing can be found in the [Chapter 3, “Messaging Server Configuration,” in Sun Java System Messaging Server 6.3 Administration Reference](#). For more information, see also:

- “[5.1.1 Enabling and Disabling Services](#)” on page 138
- “[5.1.2 Specifying Port Numbers](#)” on page 138
- “[5.2.2 Password-Based Login](#)” on page 141
- “[5.3.2 Number of Connections per Process](#)” on page 143
- “[5.3.4 Dropping Idle Connections](#)” on page 144

- “[5.3.3 Number of Threads per Process](#)” on page 144
- “[5.3.1 Number of Processes](#)” on page 142
- “[5.6.1 Configuring IMAP IDLE](#)” on page 148

Command Line: You can set values for the IMAP attributes at the command line as follows:

To enable or disable the IMAP service:

```
configutil -o service imap.enable -v [ yes | no ]
```

To specify the port number:

```
configutil -o service imap.port -v number
```

To enable a separate port for IMAP over SSL:

```
configutil -o service imap.enablesslport -v [ yes | no ]
```

To specify a port number for IMAP over SSL:

```
configutil -o service imap.sslport -v number
```

To enable or disable password login to the IMAP service:

```
configutil -o service imap.plaintextmincipher -v value
```

If *value* is > 0, then disable use of plaintext passwords unless a security layer (SSL or TLS) is activated. This forces users to enable SSL or TLS on their client to login which prevents exposure of their passwords on the network. Default is 0.

To set the maximum number of network connections per process (see “[5.3.2 Number of Connections per Process](#)” on page 143 for additional information):

```
configutil -o service imap.maxsessions -v number
```

To set the maximum idle time for connections (see “[5.3.4 Dropping Idle Connections](#)” on page 144 for additional information):

```
configutil -o service imap.idletimeout -v number
```

To set the maximum number of threads per process (see “[5.3.3 Number of Threads per Process](#)” on page 144):

```
configutil -o service imap.maxthreads -v number
```

To set the maximum number of processes (see “[5.3.1 Number of Processes](#)” on page 142):

```
configutil -o service imap.numprocesses -v number
```

To specify a protocol welcome banner:

```
configutil -o service imap.banner -v banner
```

5.6.1

Configuring IMAP IDLE

The IMAP IDLE extension to the IMAP specification, defined in RFC 2177, allows an IMAP server to notify the mail client when new messages arrive and other updates take place in a user's mailbox. The IMAP IDLE feature has the following benefits:

- Mail clients do not have to poll the IMAP server for incoming messages. Eliminating client polling reduces the workload on the IMAP server and enhances the server's performance. Client polling is most wasteful when a user receives few or no messages; the client continues to poll at the configured interval, typically every 5 or 10 minutes.
- A mail client displays a new message to the user much closer to the actual time it arrives in the user's mailbox. A change in message status is also displayed in near-realtime. The IMAP server does not have to wait for the next IMAP polling message before it can notify the client of a new or updated mail message. Instead, the IMAP server receives a notification as soon as a new message arrives or a message changes status. The server then notifies the client through the IMAP protocol.

5.6.1.1

Prerequisites

The IMAP IDLE feature relies on the Event Notification Service (ENS) to propagate notifications. To use IMAP IDLE, you must configure the following ENS components:

- An enpd server on at least one host
- The iBiff notification plug-in on all message store hosts

For information on configuring ENS for Messaging Server, see the *Sun Java System Communications Services Event Notification Service Guide*.

For information on configuring the iBiff notification plug-in, see “[B.1 Loading the ENS Publisher in Messaging Server](#)” on page 911.

▼ To Configure IMAP IDLE

1 Configure the enpd server to accept connections only from the hosts running the message stores.

To restrict connections to message-store hosts, set the ENS_ACCESS environment variable. The environment variable sets a list of permissions allowing access to enpd. The syntax is as follows:

```
setenv ENS_ACCESS 'allowdeny ipAddress|mask;  
allowdeny ipAddress|mask; ...'
```

where

<i>allowdeny</i>	Can be either + (to specify allow) or — (to specify deny)
<i>ipaddress</i>	Specifies a dotted-decimal IP address
<i>mask</i>	Specifies a dotted-decimal IP address mask

Examples:

The following example allows access to the local host only:

```
setenv ENS_ACCESS '+127.0.0.1|255.255.255.255'
```

The following example allows access to the local host and all IP addresses 192.168.0.* except 192.168.0.17:

```
setenv ENS_ACCESS '+192.168.0.1|255.255.255.0;+127.0.0.1|255.255.255.255; \
-192.168.0.17;255.255.255.255'
```

2 Run the configutil utility to specify the name of the host where the ENS server is running.

```
cd msg-svr-base
./configutil -o local.store.notifyplugin.enshost -v "ipaddress"
```

where *ipaddress* specifies a dotted-decimal IP address of the ENS host machine.

Example:

```
cd msg-svr-base
./configutil -o local.store.notifyplugin.enshost -v "127.0.0.1"
```

3 Specify the event key to use for notifications.

If the ENS event key (*ensEventKey*) is set to its default value, IMAP IDLE does not operate.

You must configure the *ensEventKey* value to end with %M. The string %M is a substitution code that is replaced by the name of the mailbox in which the event occurred.

Run the following configutil command:

```
./configutil -o local.store.notifyplugin.enseventkey -v "eventkey"
```

where *eventkey* is a unique identifier used by ENS. Its default value is enp://127.0.0.1/store. The host-name portion of the event key is not used to determine the host where ENS is running; it is simply part of the identifier.

Example:

```
./configutil -o local.store.notifyplugin.enseventkey -v "enp://127.0.0.1/store/%M"
```

- 4 Load the libibiff notification plug-in file, which enables the ENS publisher for Messaging Server.

Run the following configutil command:

```
./configutil -o local.store.notifyplugin -v "msg-svr-base/lib/libibiff"
```

- 5 Enable notifications to be sent from all user mailboxes, not only the inbox.

By default, notifications are generated only by events that occur in the inbox. However, the IMAP IDLE RFC (2177) specifies that IDLE must notify clients whenever an event occurs in any mailbox.

To comply with the RFC, the IMAP IDLE feature requires that notifications be enabled for all mailboxes. If they are not, the IMAP server will not advertise the IDLE capability.

To configure notifications for all mailboxes, set the configutil command, noneinbox, to a value of 1:

```
./configutil -o local.store.notifyplugin.noneinbox.enable -v 1
```

where -v 1 enables notifications from all mailboxes.

- 6 Stop, then restart Messaging Server.

```
cd msg-svr-base/sbin
```

```
./stop-msg
```

```
./start-msg
```

- 7 Verify that the IMAP services now include the IDLE feature. Use telnet to connect to the IMAP host and port.

```
telnet IMAP_hostname port
```

Example:

```
telnet myhost imap
trying 192.18.01.44 ...
connected to myhost.siroe.com
```

```
* OK [CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS
CHILDREN BINARY UNSELECT SORT LANGUAGE STARTTLS IDLE XSENDER X-NETSCAPE
XSERVERINFO X-SUN-SORT X-SUN-IMAP X-ANNOTATEMORE AUTH=PLAIN]
myhost.siroe.com IMAP4 service (Sun Java(tm) System
Messaging Server 6.3-0.05 (built Feb 7 2006))
```

5.7 To Configure HTTP Services

Messaging Server supports the HTTP mail clients called Messenger Express and Communications Express. While POP and IMAP clients send mail directly to a Messaging Server MTA for routing or delivery, HTTP clients send mail to a specialized web server called the Webmail Server (also called mshttpd or Messaging Server http daemon). Depending on where the message is addressed, the Webmail Server will direct the mail to an outbound MTA for routing or to one of the backend message stores using IMAP. This is shown in [Figure 5–1](#). Note that the Communications Express Server simply routes requests to and from the Webmail Server.

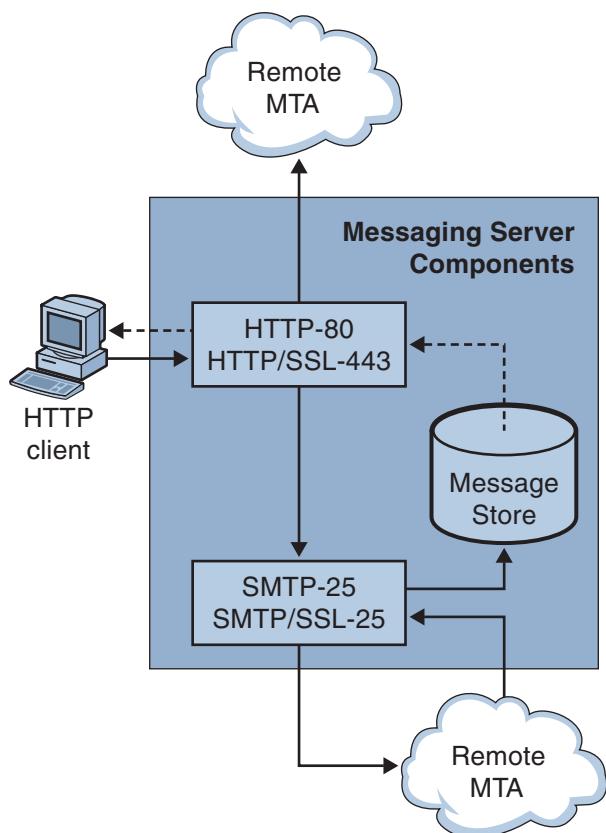


FIGURE 5–1 HTTP Service Components

In previous versions, the Webmail Server accessed the message store directly. Now, it accesses the message store through the IMAP server. This provides several advantages:

- Messenger Express and Communications Express clients are now able to access shared folders that are located on different back-end message stores.
- The Webmail Server no longer must be installed on each back-end server.
- The Webmail Server can serve as a front-end server performing the multiplexing capabilities previously performed by Messenger Express Multiplexor (MEM).
- MEM is obsoleted and is no longer used.
- On the client side, nothing is changed except that users can now access shared folders that are not on their message store.

In previous versions, the MEM received HTTP client requests and forwarded it to the appropriate Webmail Server on the back-end message store. Because of this, a copy of mshttpd had to be installed on every back-end server. Now, the Webmail Server operates as a front-end server receiving HTTP client email requests. It translates these requests to SMTP or IMAP calls and forwards the calls to either the MTA or the appropriate IMAP server on the back-end message store. If Messaging Server is used only for web-based email, make sure that IMAP is enabled.

5.7.1

Configuring Your HTTP Service

Many of the HTTP configuration parameters are similar to the parameters available for the POP and IMAP services. These include parameters for connection settings and process settings. Some of the more common HTTP service options are given in this section. A complete listing can be found in the “[configutil Parameters](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*. For more information, see also:

- “[5.1.1 Enabling and Disabling Services](#)” on page 138
- “[5.1.2 Specifying Port Numbers](#)” on page 138
- “[5.2.2 Password-Based Login](#)” on page 141
- “[5.3.2 Number of Connections per Process](#)” on page 143
- “[5.3.4 Dropping Idle Connections](#)” on page 144
- “[5.3.5 Logging Out HTTP Clients](#)” on page 145
- “[5.3.3 Number of Threads per Process](#)” on page 144
- “[5.3.1 Number of Processes](#)” on page 142

For each IMAP server that users access, the Webmail Server needs to know the IMAP port, whether to use SSL, and the admin credentials to use for user log-in. The configutil parameters to do this are as follows:

`local.service.proxy.imapport[.hostname]` — IMAP port on which to connect (default 143).

`local.service.proxy imapssl` — Enable SSL (default no).

```
local.service.proxy.admin[.hostname] — Admin ID.
```

```
local.service.proxy.adminpass[.hostname] — Admin password.
```

These parameters can be set globally (applying to every IMAP backend server), or for each individual IMAP backend server by appending the backend's fully qualified domain name to the option name.

In order to use IMAP over SSL, the mshttpd must be also configured as an SSL HTTP server, and the mshttpd certificate database must trust the IMAP backend's CA. You MUST enable service.http.sslusessl. If the backend message store running IMAP is using a self-signed certificate (for example, as created by generate-certDB) then this certificate needs to be added to the front-end mshttpd daemon server.

Note that if local.service.proxy.admin/pass isn't set, logins will be rejected with the error *Mail server unavailable. Administrator, check server log for details.* and the http log will list the missing configuration options.

Additional values for HTTP attributes can be set at the command line as follows:

To enable or disable the HTTP service:

```
configutil -o service.http.enable -v [ yes | no ]
```

By default, the HTTP service sends outgoing web mail to the local MTA for routing or delivery. You might want to configure the HTTP service to send mail to a remote MTA, for example, if your site is a hosting service and most recipients are not in the same domain as the local host machine. To send web mail to a remote MTA, you need to specify the remote host name and the SMTP port number for the remote host. To specify the port number:

```
configutil -o service.http.port -v number
```

To enable a separate port for HTTP over SSL:

```
configutil -o service.http.enablesslport -v [ yes | no ]
```

To specify a port number for HTTP over SSL:

```
configutil -o service.http.sslport -v number
```

To enable or disable password login:

```
configutil -o service.http.plaintextmincipher -v value
```

If *value* is > 0, then disable use of plaintext passwords unless a security layer (SSL or TLS) is activated. This forces users to enable SSL or TLS on their client to login which prevents exposure of their passwords on the network. Default is 0.

To set the maximum number of network connections per process (for more information, see “[5.3.2 Number of Connections per Process](#)” on page 143):

```
configutil -o service.http.maxsessions -v number
```

To set the maximum idle time for connections (for more information, see “[5.3.4 Dropping Idle Connections](#)” on page 144)

```
configutil -o service.http.idletimeout -v number
```

To set the maximum idle time for client sessions (for more information, see “[5.3.5 Logging Out HTTP Clients](#)” on page 145):

```
configutil -o service.http.sessiontimeout -v number
```

To set the maximum number of threads per process:

```
configutil -o service.http.maxthreads -v number
```

To set the maximum number of processes:

```
configutil -o service.http.numprocesses -v number
```

When an HTTP client constructs a message with attachments, the attachments are uploaded to the server and stored in a file. The HTTP service retrieves the attachments and constructs the message before sending the message to an MTA for routing or delivery. You can accept the default attachment spool directory or specify an alternate directory. You can also specify a maximum size allowed for attachments. To specify the attachment spool directory for client outgoing mail use the following command. Note that this includes all the attachments encoded in base64, and that base64 encoding requires an extra 33% more space. Thus a 5 megabyte limit in the parameter results in the maximum size of one message and attachments being about 3.75 megabytes.

```
configutil -o service.http.spooldir -v dirpath
```

To specify the maximum message size:

```
configutil -o service.http.maxmessagesize -v size
```

where *size* is a number in bytes. Note that this includes all the attachments encoded in base64, and that base64 encoding requires an extra 33% more space. Thus a 5 megabyte limit in the parameter results in the maximum size of one message and attachments being about 3.75M.

To specify an alternate MTA host name:

```
configutil -o service.http.smtphost -v hostname
```

To specify the port number for the alternate MTA host name:

```
configutil -o service.http.smtpport -v portnum
```


Enabling Single Sign-On (SSO)

Single sign-on is the ability for an end user to authenticate once (that is, log on with user ID and password) and have access to multiple applications. The Sun Java System Access Manager (note that this used to be called Identity Server) is the official gateway used for SSO for Sun Java System servers. That is, users must log into Access Manager to get access to other SSO configured servers.

For example, when properly configured, a user can sign in at the Sun Java System Access Manager login screen and have access to Messenger Express in another window without having to sign in again. Similarly, if the Sun Java System Calendar Server is properly configured, a user can sign in at the Sun Java System Access Manager login screen, then have access to their Calendar in another window without having to sign in again.

It should be noted that Messaging Server provides two methods of deploying SSO. The first way is through the Sun Java System Access Manager, the second way is through communications servers trusted circle technology. Using a trusted circle is the legacy method of implementing SSO. Though this method provides some features not available with Access Manager SSO, we do not recommend using it since all future development will be with the Access Manager. Both methods, however, are described in the following sections:

- “[6.1 Access Manager SSO for Sun Java System Servers](#)” on page 157
- “[6.2 Trusted Circle SSO \(Legacy\)](#)” on page 160

6.1 Access Manager SSO for Sun Java System Servers

This section describes SSO using Access Manager. It consists of the following sections:

- “[6.1.1 SSO Limitations and Notices](#)” on page 158
- “[6.1.2 Configuring Messaging Server to Support SSO](#)” on page 158
- “[6.1.3 Troubleshooting SSO](#)” on page 159

6.1.1

SSO Limitations and Notices

- The Messenger Express session is only valid for as long as the Access Manager session is valid. If the user logs out of Access Manager the webmail session is automatically closed (single sign-off).
- SSO Applications working together must be in the same DNS domain. (Also known as cookie domain).
- SSO Applications must have access to Access Manager verification URL (naming service).
- Browsers must have cookies.

6.1.2

Configuring Messaging Server to Support SSO

Four configutil parameters support Messaging Server SSO. Of these four, only one, local.webmail.sso.amnamingurl is required to enable SSO with Messaging Server. To enable SSO, set this parameter to the URL where Access Manager runs the naming service. Typically this URL is `http://server/amserver/namingservice`. Example:

```
configutil -o local.webmail.sso.amnamingurl -v  
http://sca-walnut:88/amserver/namingservice
```

Note – Access Manager SSO does not look at local.webmail.sso.enable which enables the older SSO mechanism. local.webmail.sso.enable should be left to off or unset otherwise warning messages will be logged about missing configuration parameters which are needed for the old SSO mechanism.

You can modify the SSO configuration parameters shown in [Table 6–3](#), by using the configutil command.

TABLE 6–1 Access Manager Single Sign-On Parameters

Parameter	Description
local.webmail.sso.amnamingurl	The URL where Access Manager runs the naming service. Mandatory variable for single sign-on through Access Manager. Typically this URL is <code>http://server/amserver/namingservice</code> . Default: Not set.

TABLE 6–1 Access Manager Single Sign-On Parameters *(Continued)*

Parameter	Description
local.webmail.sso.amcookiename	Access Manager cookie name. By default Access Manager saves its session handle in a cookie called <code>iPlanetDirectoryPro</code> . If it is configured to use another cookie name, then that name needs to be configured in Messaging Server as this parameter so that Messaging Server knows what to look for when doing single-sign on. Default value must not be changed if IS has default configuration. Default: <code>iPlanetDirectoryPro</code>
local.webmail.sso.amloglevel	AMSDK logging level. The SSO library used by Messaging Server has its own logging mechanism separate from Messaging Server. Its messages are logged in a file called <code>http_sso</code> under <code>msg-svr-base/log</code> . By default only messages with <code>info</code> or higher are logged, but it is possible to increase the logging level by setting the logging level to a value from 1 to 5 (1 = errors, 2 = warnings, 3 = info, 4 = debug, 5 = maxdebug). Be aware that the library doesn't have the same notion of message importances as Messaging Server and that setting the level to <code>debug</code> can result in a lot of meaningless data. Also the <code>http_sso</code> log file is not managed by common Messaging Server logging code and is never cleaned up or rolled over. It is the responsibility of the system administrator to clean it up when setting the log level higher than the default. Default: 3
local.webmail.sso.singlesignoff	Single sign-off from Messaging Server to Access Manager. Access Manager is the central authentication authority, and single sign-off is always enabled from Access Manager to Messaging Server. This option allows a site to configure whether the <code>logout</code> button in webmail should also log the user out of Access Manager (saving some customization work). By default this is enabled. If this is disabled, a user logging out of the default webmail client is automatically logged back in since <code>logout</code> refers to the root document and the root document refers to the inbox display as long as the Access Manager cookie exists and is valid. Therefore, a site choosing to disable this option needs to customize what happens at webmail logout. Default: yes

6.1.3 Troubleshooting SSO

If there is a problem with SSO, the first thing to do is check the webmail log file `msg-svr-base/log/http` for errors. Increasing the logging level may be helpful (`configutil -o logfile.http.loglevel -v debug`). If this does not help, then check the `amsdk` messages in `msg-svr-base/log/http_sso`, then increase the `amsdk` logging level (`configutil -o local.webmail.sso.amloglevel -v 5`). Note that new logging levels only take effect after server restart.

If SSO is still having problems, make sure you are using fully qualified host names of both Access Manager and Messaging Server during log in. Cookies are only shared between servers

of the same domain, and browsers do not know what the domain is for local server names, so one must use the fully qualified names in the browser for SSO to work.

6.2 Trusted Circle SSO (Legacy)

This section describes trusted circle SSO. We do not recommend using this method of SSO since all future development will be with the Access Manager. However, there is some functionality available with trusted circle SSO that is not available with Access Manager SSO at this time. This section consists of the following sections:

- “[6.2.1 Trusted Circle SSO Overview and Definitions](#)” on page 160
- “[6.2.2 Trusted Circle SSO Applications](#)” on page 161
- “[6.2.3 Trusted Circle SSO Limitations](#)” on page 161
- “[6.2.4 Example Trusted Circle SSO Deployment Scenarios](#)” on page 161
- “[6.2.5 Setting Up Trusted Circle SSO](#)” on page 163
- “[6.2.6 Messenger Express Trusted SSO Configuration Parameters](#)” on page 167

6.2.1

Trusted Circle SSO Overview and Definitions

Before deploying SSO it is important to understand the following terminology.

- **SSO:** Single Sign-On. The ability to sign on to one application and be able access the other applications. The user identification is the same throughout all applications.
- **Trusted applications.** Applications sharing the SSO scheme (*SSO Prefix*) and trusting each other’s cookies and verifications. Also known as *Peer SSO applications*.
- **Trusted circle.** The circle of trusted applications. They share the same SSO Prefix.
- **SSO Prefix.** A string defined by the person deploying SSO and made known to applications so they can use it to find cookies generated by other applications in the same trusted circle. Applications with different prefixes are not in the same circle and the user needs to re-authenticate when moving between these applications. The prefix sometimes, but not always, explicitly contains the trailing - (“-”) in the configuration setting.
- **Application ID.** (appid). A unique string defined by the person deploying SSO for each application in the SSO circle.
- **SSO Cookie.** A token that the browser uses to remember that the user has authenticated to some application. The name of the cookie is of the form *SSO_prefix-application ID*. The value of the cookie is the SSO key, usually a session ID generated by the application.
- **Cookie Domain.** A domain within which the application is restricted to send cookies. This is a domain in the DNS sense.
- **Verification URL.** A URL used by one application to verify the cookie it found to another application.

6.2.2

Trusted Circle SSO Applications

Before implementing SSO, you must first consider which applications will be in this trusted circle. The applications which can be in this trusted circle are Messenger Express, Calendar Express, and the old iPlanet Delegated Administrator for Messaging (not recommended because it only supports Sun LDAP Schema 1).

[Table 6–2](#) shows which applications can be accessed from each other via SSO. From a user's point of view, logging into one of the applications on the first column, SSO works if the user is able to access application across the top row without having to re-enter user id and passwords.

TABLE 6–2 SSO Interoperability

To:	Calendar Express	Messenger Express	Delegated Administrator
From:			
Calendar Express	SSO	SSO	SSO
Messenger Express	SSO	N/A	SSO
Delegated Administrator	SSO	SSO	N/A

6.2.3

Trusted Circle SSO Limitations

- SSO Applications working together must be in the same domain.
- SSO Applications must have access to each other's SSO verification URL.
- Browsers must support cookies.
- For security purposes, SSO should not be used on machines where the browser is run.
- To switch to a different identity, the browser needs to be restarted.
- Assuming single sign-off is enabled in both Messenger Express and for Sun Java System Calendar Server, if you log out of Sun Java System Calendar Server, then you are supposed to have to login again to Messenger Express. If you log out of Messenger Express, then you would have to login again into Sun Java System Calendar Server. However, it currently does not work this way. You may stay logged into one after logging out of another.

6.2.4

Example Trusted Circle SSO Deployment Scenarios

The simplest SSO deployment scenario consists of only Messenger Express and Delegated Administrator. A more complicated scenario can be created by adding Calendar Express—on the same machine or a different machine—using the same SSO prefix so they are in the same trusted circle. This is shown in [Figure 6–1](#).

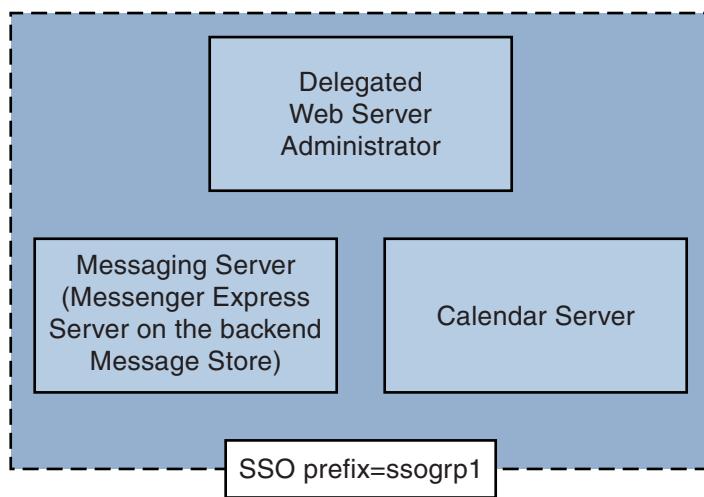


FIGURE 6-1 Simple SSO Deployment

An even more complex deployment would include Webmail Servers and load balancers.

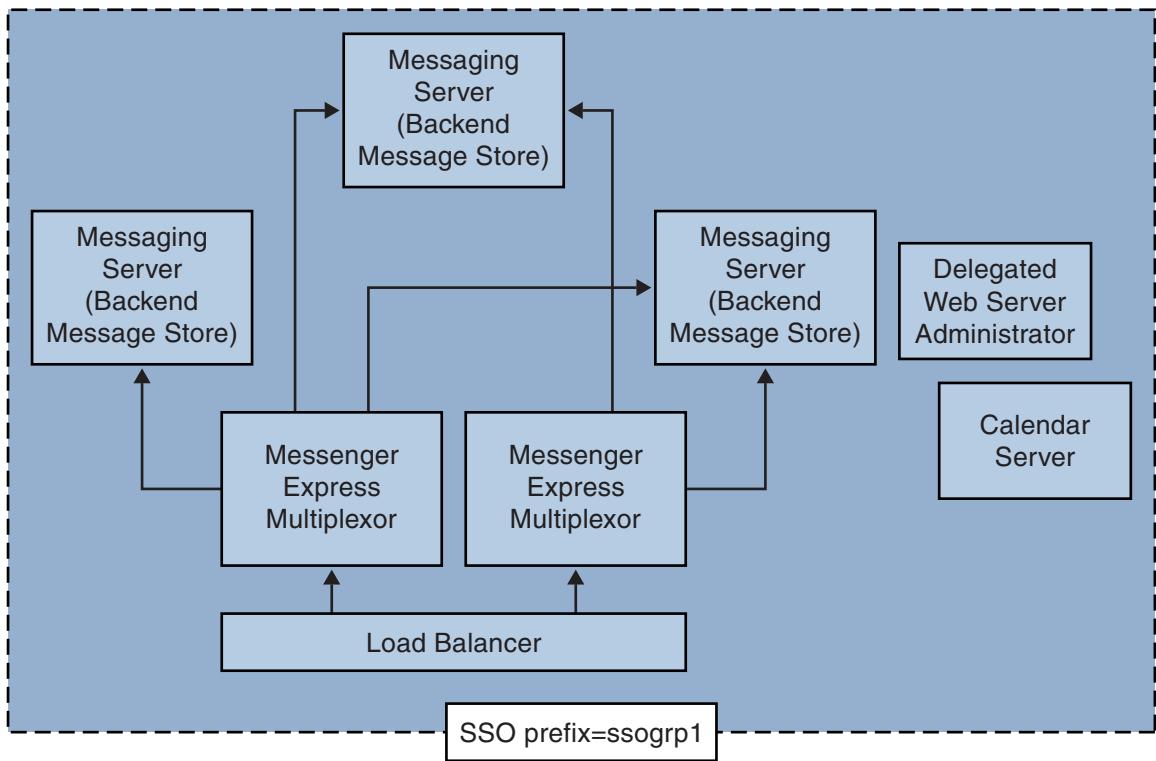


FIGURE 6-2 Complex SSO Deployment

6.2.5 Setting Up Trusted Circle SSO

This section describes setting up SSO for Messenger Express, Delegated Administrator, and Calendar Manager.

▼ To Set Up SSO for Messenger Express, Delegated Administrator, and Calendar Manager

1 Configure Messenger Express for SSO.

a. Set the appropriate SSO configutil parameters.

To enable single sign-on for Messenger Express with Delegated Administrator, set the configuration parameters as follows (assumes your default domain is siroe.com). These parameters are described in Table 6-3. You must be root user. cd to *instance_root*

```
configutil -o local.webmail.sso.enable -v 1
configutil -o local.webmail.sso.prefix -v ssogrp1
```

`ssogrp1` is the default SSO Prefix used by Delegated Administrator, although you can choose a different prefix, using the default would save a little typing when configuring Delegated Administrator and Calendar Server.

```
configutil -o local.webmail.sso.id -v ims5
```

`ims5` is a name you pick to identify Messenger Express (ME) to other applications.

```
configutil -o local.webmail.sso.cookieDomain -v ".siroe.com"
```

The above domain must match the domain used by the ME/browser client to connect to the servers. Thus, although the hosted domain on this server may be called `xyz.com`, we must use a real domain in the DNS. This value must start with a period.

```
configutil -o local.webmail.sso.singleSignoff -v 1  
configutil -o local.sso.ApplicationID.verifyurl -v \  
"http://ApplicationHost:port/VerifySSO?"
```

`ApplicationID` is a name we give to the SSO application (example: `ida` for Delegated Administrator, `ics50` for Calendar Server). `ApplicationHost:port` is the host and port number of the application. You will have one of these lines for each non-Messaging Server application. Example:

```
configutil -o local.sso.ida.verifyurl -v \  
"http://siroe.com:8080/VerifySSO?"
```

b. Restart Messenger Express http server after changing the configuration.

```
cd instance_root./stop-msg http  
.start-msg http
```

2 Configure Directory Server for SSO.

a. Create a proxy user account in the directory.

The proxy user account allows the Delegated Administrator to bind to the Directory Server for proxy authentication. Using the following LDIF code (`proxy.ldif`) you could create a proxy user account entry using `ldapadd`.

```
ldapadd -h mysystem.siroe.com -D "cn=Directory Manager" -w password -v -f  
proxy.ldif  
dn: uid=proxy, ou=people, o=siroe.com, o=isp  
objectclass: top  
objectclass: person  
objectclass: organizationalperson  
objectclass: inetorgperson  
uid: proxy  
givenname: Proxy  
sn: Auth
```

```
cn: Proxy Auth
userpassword: proxypassword
```

b. Create the appropriate ACIs for proxy user account authentication.

Using the `ldapmodify` utility, create an ACI for each of the suffixes you created at the time you installed the Delegated Administrator.

`osiroot` - The suffix you entered to store the user data (the default is `o=isp`). `osiroot` is the root of the Organization Tree.

`dcroot` - The suffix you entered to store the domain information. (The default is `o=internet`.)

`osiroot` - The suffix you entered to store the configuration information, it should have been the same value you entered to store the user data.

The following is an example of an ACI entry (`aci1.ldif`) for the `osiroot` for the proxy user created earlier:

```
dn: o=isp
changetype: modify
add: aci
aci: (target="ldap:///o=isp")(targetattr="*")(version 3.0; acl
"proxy";allow (proxy) userdn="ldap:///uid=proxy, ou=people,
o=siroe.com, o=isp";)
```

```
ldapmodify -h siroe.com -D "cn=Directory Manager" -w password -v
-f aci1.ldif
```

Create a similar ACI entry (`aci2.ldif`) for the `dcroot`:

```
dn: o=internet
changetype: modify
add: aci
aci: (target="ldap:///o=internet")(targetattr="*")(version 3.0; acl
"proxy";allow (proxy) userdn="ldap:///uid=proxy, ou=people,
o=siroe.com, o=isp";)
```

```
ldapmodify -h siroe.com -D "cn=Directory Manager" -w password -v
-f aci2.ldif
```

3 Configure the Delegated Administrator

a. Add the proxy user credentials and cookie name for context to the Delegated Administrator `resource.properties` file.

Uncomment and modify the following entries in the Delegated Administrator `resource.properties` file:

```
LDAPDatabaseInterface-ldapauthdn=Proxy_Auth_DN
LDAPDatabaseInterface-ldapauthpw=Proxy_Auth_Password
```

```
NDAAuth-singleSignOnId=SSO_Prefix-
NDAAuth-applicationId=DelAdminID
```

For example:

```
LDAPDatabaseInterface-ldapauthdn= uid=proxy,ou=people,o=cesta.com,o=isp
LDAPDatabaseInterface-ldapauthpw=proxypassword
NDAAuth-singleSignOnId=ssogrp1-
NDAAuth-applicationId=ida
```

The resource.properties file is stored in the following location:

iDA_svr_base/nda/classes/netscape/nda/servlet/

b. Add the participating server's verification URL.

To verify a single sign-on cookie it receives, Delegated Administrator must know who to contact. You must provide a verification URL for all known participating servers.

Following the example, assume Messenger Express is installed and its application ID is msg5. Edit the Delegated Administrator resource.properties file and add an entry such as:

```
verificationurl-ssogrp1-msg5=http://webmail_hostname:port/VerifySSO?
verificationurl-ssogrp1-ida=http://iDA_hostname:port/VerifySSO?
verificationurl-ssogrp1-ics50=http://iCS_hostname:port/VerifySSO?
```

The resource.properties file is located in the following directory:

iDA_svr_base/nda/classes/netscape/nda/servlet/

4 Add Delegated Administrator single sign-on cookie information and enable UTF8 Parameter Encoding.

a. Define a context identifier for Delegated Administrator.

Edit the servlets.properties file and uncomment all lines containing the text `servlet.*.context=ims50`. Where * is any string.

The servlets.properties file is located at:

Web_Svr_Base/https-instancename/config/

b. Specify a cookie name for the context in the Enterprise Server configuration.

Edit the Enterprise Server contexts.properties file and add the following line to the bottom of the file before the #IDACONF-Start line:

```
context.ims50.sessionCookie=ssogrp1-ida
```

The contexts.properties file is located at:

Web_Svr_Base/https-instancename/config/

c. Enable UTF8 parameter encoding for ims5 contexts.

To Enable UTF8 Parameter Encoding for ims5 Contexts in the Enterprise Server configuration add the following entry to the Enterprise Server contexts.properties file:

```
context.ims50.parameterEncoding=utf8
```

5 Restart Messenger Express.

After you've made the configuration changes described in steps 1a through 2c, you must restart Messenger Express for the changes to take effect:

```
Web_Svr_Base/https-instance_name/stop  
Web_Svr_Base/https-instancename/start
```

6 If you are deploying Calendar in this SSO group, configure Calendar Server.

Edit ics.conf and add the following:

```
sso.appid = "ics50"  
sso.appprefix = "ssogrp1"  
sso.cookieDomain = ".red.iplanet.com"  
sso.enable = "1"  
sso.singleSignoff = "true"  
sso.userDomain = "mysystem.red.iplanet.com"  
sso.ims5.url="http://mysystem.red.iplanet.com:80/VerifySSO?"  
sso.ida.url=http://mysystem.red.iplanet.com:8080/VerifySSO?
```

7 Restart Calendar Server

```
start-cal
```

8 Restart the Messenger Express http server:

```
msg-svr-base/sbin/stop-msg http  
msg-svr-base/sbin/start-msg http
```

6.2.6

Messenger Express Trusted SSO Configuration Parameters

You can modify the single sign-on configuration parameters for Messenger Express, shown in “[6.2.6 Messenger Express Trusted SSO Configuration Parameters](#)” on page 167, by using the configutil command. For more information about configutil, see the *Sun Java System Messaging Server 6.3 Administration Reference*

TABLE 6-3 Trusted Circle Single Sign-On Parameters

Parameter	Description
<code>local.sso.appid.verifyurl</code>	<p>Sets the verify URL values for peer SSO applications. <i>appid</i> is the application ID of a peer SSO application whose SSO cookies are to be honored. For example, the default <i>appid</i> for Delegated Administrator is <code>nda45</code>. Its actual value is specified by the Delegated Administrator <code>resource.properties</code> file entry <code>NDAAuth-applicationID</code>.</p> <p>There should be one parameter defined for each trusted peer SSO application. The standard form of the verify URL is:</p> <pre>http://nda-host:port/VerifySSO?</pre> <p>If you are using a load balancer in front of multiple Webmail Servers and Message Store servers (running Messenger Express) or Calendar front ends, be sure to assign a <i>different</i> <i>appid</i> for each physical system with the real host names in the <code>verifyurl</code>. This will ensure that the correct system will be used to verify the cookie</p>
<code>local.webmail.sso.cookiedomain</code>	<p>The string value of this parameter is used to set the cookie domain value of all SSO cookies set by the Messenger Express HTTP server. The default value is null.</p> <p>This domain must match the DNS domain used by the Messenger Express browser to access the server. It is not the hosted domain name.</p>
<code>local.webmail.sso.enable</code>	<p>Enables or disables all single sign-on functionality, including accepting and verifying SSO cookies presented by the client when the login page is fetched, returning an SSO cookie to the client on successful login and responding to requests from other SSO partners to verify its own cookies.</p> <p>If set to any non-zero value, the server performs all SSO functions.</p> <p>If set to zero, the server does not perform any of these SSO functions.</p> <p>The default value is zero.</p>
<code>local.webmail.sso.id</code>	<p>The string value of this parameter is used as the application ID value when formatting SSO cookies set by the Messenger Express HTTP server. The default value is null.</p> <p>This is an arbitrary string. Its value must match what you specify for the Delegated Administrator in its <code>resource.properties</code> file. The corresponding entry in <code>resource.properties</code> would be:</p> <pre>Verifcationurl-XXX-YYY=http://webmailhost: webmailport/VerifySSO?</pre> <p>Where XXX is the <code>local.webmail.sso.prefix</code> value set above, and YYY is the value of <code>local.webmail.sso.id</code> set here.</p>

TABLE 6–3 Trusted Circle Single Sign-On Parameters *(Continued)*

Parameter	Description
<code>local.webmail.sso.prefix</code>	<p>The string value of this parameter is used as the prefix value when formatting SSO cookies set by the Messenger Express HTTP server. Only SSO cookies with this prefix will be recognized by the server; all other SSO cookies will be ignored.</p> <p>A null value for this parameter effectively disables all SSO functionality on the server.</p> <p>The default value is null.</p> <p>This string must match what is used by the Delegated Administrator in its <code>resource.properties</code> file without the trailing <code>-</code>. For example, if:</p> <pre>NDAAAuth-singleSignOnID=ssogrp1-</pre> <p>Then this value should be set here to <code>ssogrp1</code>.</p>
<code>local.webmail.sso.singleSignoff</code>	<p>The integer value of this parameter, if set to any non-zero value, clears all SSO cookies on the client with prefix values matching the value configured in <code>local.webmail.sso.prefix</code> when the client logs out.</p> <p>If set to zero, Messenger Express will clear its own SSO cookie when the client logs out.</p> <p>The default value is zero.</p>

Configuring and Administering Multiplexor Services

This chapter describes the Messaging Multiplexor (MMP) for standard mail protocols (POP, IMAP and SMTP). Previous releases also described the Messenger Express Multiplexor used for the Messenger Express web interface, but this is no longer needed. See “[5.7 To Configure HTTP Services](#)” on page 151.

The following topics are covered in this chapter:

- “[7.1 Multiplexor Services](#)” on page 171
- “[7.2 About Messaging Multiplexor](#)” on page 173
- “[7.3 Setting Up the Messaging Multiplexor](#)” on page 178
- “[7.4 Configuring MMP with SSL](#)” on page 181
- “[7.5 MMP Tasks](#)” on page 186

7.1 Multiplexor Services

A multiplexor is necessary to achieve horizontal scalability (the ability to support more users by adding more machines), because it provides a single domain name that can be used to connect indirectly to multiple mail stores. A multiplexor can also provide security benefits.

While MMP is managed separately from Messaging Server, the Messenger Express multiplexing is built-in to the HTTP service (`mshttpd`) that is included with the Message Store and Message Access installation.

7.1.1 Multiplexor Benefits

Message stores on heavily used messaging servers can grow quite large. Spreading user mailboxes and user connections across multiple servers can therefore improve capacity and performance. In addition, it may be more cost-effective to use several small server machines than one large, high-capacity, multiprocessor machine.

If the size of your mail-server installation requires the use of multiple message stores, your organization can benefit in several ways from using the multiplexor. The indirect connection between users and their message stores, coupled with the ease of reconfiguration of user accounts among messaging servers allows for the following benefits:

- **Simplified User Management**

Because all users connect to one server (or more, if you have separate Multiplexor machines for POP, IMAP, SMTP or web access), you can preconfigure email clients and distribute uniform login information to all users. This simplifies your administrative tasks and reduces the possibility of distributing erroneous login information.

For especially high-load situations, you can run multiple Multiplexor servers with identical configurations and manage connections to them by DNS round robin or by using a load-balancing system.

Because the Multiplexors use information stored in the LDAP directory to locate each user's Messaging Server, moving a user to a new server is simple for the system administrator and transparent to the user. The administrator can move a user's mailbox from one Messaging Server to another, and then update the user's entry in the LDAP directory. The user's mail address, mailbox access, and other client preferences need not change.

- **Improved Performance**

If a message store grows prohibitively large for a single machine, you can balance the load by moving some of the message store to another machine.

You can assign different classes of users to different machines. For example, you can choose to locate premium users on a larger and more powerful machine.

The Multiplexors perform some buffering so that slow client connections (through a modem, for example) do not slow down the Messaging Server.

- **Decreased Cost.** Because you can efficiently manage multiple Messaging Servers with a Multiplexor, you might be able to decrease overall costs by purchasing several small server machines that together cost less than one very large machine.
- **Better Scalability.** With the Multiplexors, your configuration can expand easily. You can incrementally add machines as your performance or storage-capacity needs grow, without replacing your existing investment.
- **Minimum User Downtime.** Using the Multiplexors to spread a large user base over many small store machines isolates user downtime. When an individual server fails, only its users are affected.
- **Increased Security.** You can use the server machine on which the Multiplexor is installed as a firewall machine. By routing all client connections through this machine, you can restrict access to the internal message store machines by outside computers. The Multiplexors support both unencrypted and encrypted communications with clients.

7.2 About Messaging Multiplexor

The Sun Java System Messaging Multiplexor (MMP) is a specialized messaging server that acts as a single point of connection to multiple back-end messaging servers. With Messaging Multiplexor, large-scale messaging service providers can distribute POP and IMAP user mailboxes across many machines to increase message store capacity. All users connect to the single Multiplexor server, which redirects each connection to the appropriate messaging server.

If you provide electronic mail service to many users, you can install and configure the Messaging Multiplexor so that an entire array of messaging servers will appear to your mail users to be a single host.

The Messaging Multiplexor is provided as part of Messaging Server. You can install the MMP at the same time you install the Messaging Server or other Sun Java System servers, or you can install the MMP separately at a later time. The MMP supports:

- Both unencrypted and encrypted (SSL) communications with mail clients.
- Client certificate-based authentication, described in “[7.2.3 Certificate-Based Client Authentication](#)” on page 175
- User pre-authentication, described in “[7.2.4 User Pre-Authentication](#)” on page 176.
- Virtual domains that listen on different IP addresses and automatically append domain names to user IDs, described in “[7.2.5 MMP Virtual Domains](#)” on page 176.
- Multiple installations of the MMP on different servers
- Enhanced LDAP searching.
- POP before SMTP service for legacy POP clients. For more information, see “[23.8 Enabling POP Before SMTP](#)” on page 746.

This section consists of the following subsections:

- “[7.2.1 How the Messaging Multiplexor Works](#)” on page 173
- “[7.2.2 Encryption \(SSL\) Option](#)” on page 175
- “[7.2.3 Certificate-Based Client Authentication](#)” on page 175
- “[7.2.4 User Pre-Authentication](#)” on page 176
- “[7.2.5 MMP Virtual Domains](#)” on page 176
- “[7.2.6 About SMTP Proxy](#)” on page 178

7.2.1

How the Messaging Multiplexor Works

The MMP is a multithreaded server that facilitates distributing mail users across multiple server machines. The MMP handles incoming client connections destined for other server machines (the machines on which user mailboxes reside). Clients connect to the MMP itself, which determines the correct server for the users, connects to that server, and then passes data between the client and server. This capability allows Internet service providers and other large

installations to spread message stores across multiple machines (to increase capacity) while providing the appearance of a single mail host for users (to increase efficiency) and for external clients (to increase security). “[7.2.1 How the Messaging Multiplexor Works](#)” on page 173 shows how servers and clients relate to each other in an MMP installation.

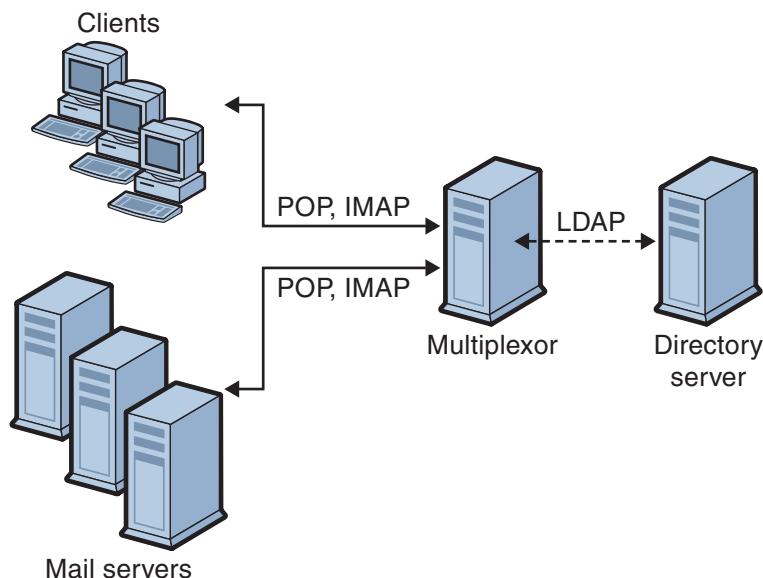


FIGURE 7-1 Clients and Servers in an MMP Installation

All POP, IMAP, and SMTP clients work with the Messaging Multiplexor. The MMP accepts connections, performs LDAP directory lookups, and routes the connections appropriately. As is typical with other mail server installations, each user is assigned a specific address and mailbox on a specific Messaging Server. However, all connections are routed through the MMP.

In more detail, these are the steps involved in establishing a user connection:

1. A user's client connects to the MMP, which accepts preliminary authentication information (user name).
2. The MMP queries the Directory Server to determine which Messaging Server contains that user's mailbox.
3. The MMP connects to the proper Messaging Server, replays authentication, then acts as a pass-through pipe for the duration of the connection.

7.2.2

Encryption (SSL) Option

Messaging Multiplexor supports both unencrypted and encrypted (SSL) communications between the Messaging Server(s) and their mail clients. The current version of Messaging Server supports the new certificate database format (cert8.db).

When SSL is enabled, the MMP supports STARTTLS and the MMP can also be configured to listen on additional ports for SSL IMAP, POP, and SMTP connections.

To enable SSL encryption for your IMAP, POP, and SMTP services, edit the `ImapProxyAService.cfg`, `PopProxyAService.cfg`, and `SsmtpProxyAService.cfg` files, respectively. You must also edit the `default:ServiceList` option in the `AService.cfg` file to include the list of all IMAP, POP, and SMTP server ports regardless of whether or not they are secure. See “[7.4 Configuring MMP with SSL](#)” on page 181 for details.

By default, SSL is not enabled since the SSL configuration parameters are commented out. To enable SSL, you must install an SSL server certificate. Then, you should uncomment and set the SSL parameters. For a list of the SSL parameters, see the “[Encryption \(SSL\) Option](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

7.2.3

Certificate-Based Client Authentication

The MMP can use a certificate mapping file (`certmap.conf`) to match a client’s certificate to the correct user in the Users/Groups Directory Server.

In order to use certificate-based client authentication, you must also enable SSL encryption as described in “[7.2.2 Encryption \(SSL\) Option](#)” on page 175.

You also have to configure a store administrator. You can use the mail administrator, but it is recommended that you create a unique user ID, such as `mmpstore` for this purpose so that you can set permissions as needed.

Note that the MMP does not support `certmap` plug-ins. Instead, the MMP accepts enhanced `DNComps` and `FilterComps` property value entries in the `certmap.conf` file. These enhanced format entries use the form:

`mapname:DNComps FROMATTR=TOATTR mapname:FilterComps FROMATTR=TOATTR`

So that a `FROMATTR` value in a certificate’s subjectDN can be used to form an LDAP query with the `TOATTR=value` element. For example, a certificate with a subjectDN of “`cn=Pilar Lorca, ou=pilar, o=siroe.com`” could be mapped to an LDAP query of “`(uid=pilar)`” with the line:

`mapname:FilterComps ou=uid`

▼ To Enable Certificate-based Authentication for Your IMAP or POP Service

- 1 Decide on the user ID you intend to use as store administrator.
While you can use the mail administrator for this purpose, it is recommended that you create a unique user ID for store administrator (for example, `mmpstore`).
- 2 Make sure that SSL encryption is (or will be) enabled as described in “[7.2.2 Encryption \(SSL Option\)](#)” on page 175.
- 3 Configure the MMP to use certificate-based client authentication by specifying the location of the `certmap.conf` file in your configuration files.
- 4 Install at least one trusted CA certificate, as described in “[23.5.1.6 To Install Certificates of Trusted CAs](#)” on page 726

7.2.4 User Pre-Authentication

The MMP provides you with the option of pre-authenticating users by binding to the directory as the incoming user and logging the result.

Note – Enabling user pre-authentication will reduce server performance

The log entries are in the format:

```
date time (sid 0xhex) user name pre-authenticated - client  
IP address, server IP address
```

Where *date* is in the format `yyyymmdd`, *time* is in the time configured on the server in the format `hhmmss`, *hex* is the session identifier (*sid*) represented as a hexadecimal number, the user *name* includes the virtual domain (if any), and the IP address is in dot-quad format.

7.2.5 MMP Virtual Domains

An MMP virtual domain is a set of configuration settings associated with a server IP address. The primary use of this feature is to provide different default domains for each server IP address.

A user can authenticate to MMP with either a short-form userID or a fully qualified userID in the form `user@domain`. When a short-form userID is supplied, the MMP will append the `DefaultDomain` setting, if specified. Consequently, a site which supports multiple hosted

domains can permit the use of short-form user IDs simply by associating a server IP address and MMP virtual domain with each hosted domain.

The recommended method for locating the user subtree for a given hosted domain is via the `inetDomainBaseDN` attribute in the LDAP domain tree entry for that domain. The MMP's `LdapUrl` setting is not suitable for this purpose since the back-end mail store servers will also need to look up the user in LDAP and do not support virtual domains.

When Sun LDAP Schema 2 is enabled (see the *Sun Java Enterprise System 5 Installation Guide for UNIX* and *Sun Java Communications Suite 5 Schema Reference*), the user subtree for the specified domain will be all the users in the subtree below the organization node for that domain.

To enable virtual domains, edit the `ImapProxyAService.cfg`, `PopProxyAService.cfg`, or `SmtpProxyAService.cfg` file(s) in the instance directory such that the `VirtualDomainFile` setting specifies the full path to the virtual domain mapping file.

Each entry of a virtual domain file has the following syntax:

```
vdmap name IPaddr  
name:parameter value
```

Where `name` is simply used to associate the IP address with the configuration parameters and can be any name you choose to use, `IPaddr` is in dot-quad format, and `parameter` and `value` pairs configure the virtual domain. When set, virtual domain configuration parameter values override global configuration parameter values.

Listed below are the configuration parameters you can specify for a virtual domain:

```
AuthCacheSize and AuthCacheSizeTTL  
AuthService  
BindDN and BindPass  
CertMap  
ClientLookup  
CRAMs  
DefaultDomain  
DomainDelim  
HostedDomains  
LdapCacheSize and LdapCacheTTL  
LdapURL  
MailHostAttrs  
PreAuth  
ReplayFormat  
RestrictPlainPasswords  
StoreAdmin and StoreAdminPass  
SearchFormat  
TCPAccess  
TCPAccessAttr
```

Note – Unless the LdapURL is correctly set, the BindDN, BindPass, LdapCacheSize and LdapCacheTTL settings will be ignored.

For detailed descriptions of these configuration parameters, see the *Sun Java System Messaging Server 6.3 Administration Reference*

7.2.6

About SMTP Proxy

The MMP includes an SMTP proxy which is disabled by default. Most sites do not need the SMTP proxy because Internet Mail standards already provide an adequate mechanism for horizontal scalability of SMTP (DNS MX records).

The SMTP proxy is useful for the security features it provides. First, the SMTP proxy is integrated with the POP proxy to implement the POP before SMTP authorization facility required by some legacy POP clients. For more information, see “[Using the MMP SMTP Proxy](#)” in *Sun Java Communications Suite 5 Deployment Planning Guide* and “[23.8 Enabling POP Before SMTP](#)” on page [746](#). In addition, an investment in SSL acceleration hardware can be maximized by using the SMTP proxy. See “[23.5.4 How to Optimize SSL Performance Using the SMTP Proxy](#)” on page [734](#).

7.3 Setting Up the Messaging Multiplexor

During the initial runtime configuration of Messaging Server, you determined if you wanted to configure the MMP on a machine. You could either set it up on the same machine as your Messaging Server or set it up on a separate machine.

Note – MMP does not cache DNS results. A high quality caching DNS server on the local network is a requirement for a production deployment of Messaging Server.

The following sections describe how to set up the MMP:

- [“7.3.1 Before You Configure MMP” on page 179](#)
- [“7.3.2 Multiplexor Configuration” on page 179](#)
- [“7.3.3 Multiplexor Files” on page 180](#)
- [“7.3.4 Starting the Multiplexor” on page 181](#)
- [“7.3.5 Modifying an Existing MMP” on page 181](#)

More information about the MMP can be found in the following:

- [Chapter 5, “Messaging Multiplexor Configuration,” in *Sun Java System Messaging Server 6.3 Administration Reference*](#)

7.3.1

Before You Configure MMP

Before configuring the MMP:

1. Choose the machine on which you will configure the MMP. It is best to use a dedicated machine set aside for the MMP.

Note – It is recommended that the MMP not be enabled on a machine that is also running either the POP or IMAP servers.

If you install MMP on the same machine as Messaging Server, you must make sure that the POP and IMAP servers are set to non-standard ports. That way, the MMP and Messaging Server ports will not conflict with one another.

2. On the machine that the MMP is to be configured, create a UNIX system user to be used by the MMP. This new user must belong to a UNIX system group. See “[1.1 Creating UNIX System Users and Groups](#)” on page 63
3. Set up the Directory Server and its host machine for use with Messaging Server, if they are not already set up. See “[1.2 To Prepare Directory Server for Messaging Server Configuration](#)” on page 64
4. If the MMP is upgraded before the backend servers, customers should set the Capability option in `ImapProxyAService.cfg` to match the response to the capability command from the older backend. This would be:

```
IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS CHILDREN LANGUAGE  
XSENDER X-NETSCAPE XSERVERINFO
```

Note that line break is for editorial clarity, and that the configuration value must be on one line.

7.3.2

Multiplexor Configuration

To configure the MMP, you must use the Messaging Server configure program, which gives you the option of enabling the Messaging Multiplexor. For detailed information about the configure program see “[1.3 Creating the Initial Messaging Server Runtime Configuration](#)” on page 65

▼ To Configure the MMP

1. **Put Sun Java System Messaging Server on the machine where you are installing and configuring the MMP.**

- 2 Configure the MMP by creating the Messaging Server Initial Runtime Configuration. See “[1.3 Creating the Initial Messaging Server Runtime Configuration](#)” on page [65](#)

Note the following exception: when installing Messaging Server, check only the Messaging Multiplexor option.

7.3.3 Multiplexor Files

The Messaging Multiplexor files are stored in the *msg-svr-base/config* configuration file directory. You must manually edit the configuration parameters in the Messaging Multiplexor configuration files listed in Table 7–1. For a complete description of all MMP configuration parameters, see the “[Multiplexor Configuration Parameters](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

TABLE 7–1 Messaging Multiplexor Configuration Files

File	Description
<code>PopProxyAService.cfg</code>	Configuration file specifying configuration variables used for POP services.
<code>PopProxyAService-def.cfg</code>	POP services configuration template. File comes into existence only after initial MMP start with <code>start-msg mmp</code>
<code>ImapProxyAService.cfg</code>	Configuration file specifying configuration variables used for IMAP services.
<code>ImapProxyAService-def.cfg</code>	IMAP services configuration template. File comes into existence only after initial MMP start with <code>start-msg mmp</code>
<code>AService.cfg</code>	Configuration file specifying which services to start and a few options shared by both POP and IMAP services.
<code>AService-def.cfg</code>	Configuration template specifying which services to start and a few options shared by both POP and IMAP services. File comes into existence only after initial MMP start with <code>start-msg mmp</code>
<code>SmtpProxyAService.cfg</code>	Optional configuration file specifying configuration variables used for SMTP Proxy services. Required if you enable POP before SMTP; useful for maximizing support for SSL hardware even if POP before SMTP is not enabled. For more information on POP before SMTP, see “ 23.8 Enabling POP Before SMTP ” on page 746
<code>SmtpProxyAService-def.cfg</code>	Configuration template specifying configuration variables used for SMTP Proxy services. File comes into existence only after initial MMP start with <code>start-msg mmp</code>

As an example, the `LogDir` and `LogLevel` parameters can be found in all configuration files. In `ImapProxyAService.cfg`, they are used to specify logging parameters for IMAP-related events; similarly, these parameters in `PopProxyAService.cfg` are used to configure logging parameters for POP-related events. In `SmtpProxyAService.cfg`, they are used to specify logging for SMTP Proxy-related events.

In `AService.cfg`, however, `LogDir` and `LogLevel` are used for logging MMP-wide failures, such as the failure to start a POP, IMAP, or SMTP service.

Note – When the MMP is configured or upgraded, the configuration template files will be overwritten.

7.3.4 Starting the Multiplexor

To start, stop, or refresh an instance of the Messaging Multiplexor, use the one of the following commands in [Table 7–2](#) located in the `msg-svr-base/sbin` directory:

TABLE 7–2 MMP Commands

Option	Description
<code>start-msg mmp</code>	Starts the MMP (even if one is already running).
<code>stop-msg mmp</code>	Stops the most recently started MMP.
<code>refresh mmp</code>	Causes an MMP that is already running to refresh its configuration without disrupting any active connections.

7.3.5 Modifying an Existing MMP

To modify an existing instance of the MMP, edit the `ImapProxyAService.cfg` and/or `PopProxyAService.cfg` configuration files as necessary. These configuration files are located in the `msg-svr-base/config` subdirectory.

7.4 Configuring MMP with SSL

To configure the MMP to use SSL, do the following:

Note – It is assumed that the MMP is installed on a machine that does not have a Message Store or MTA.

▼ To Configure MMP with SSL

- 1 [Install an SSL server certificate \(see “23.5 Configuring Encryption and Certificate-Based Authentication” on page 720.](#)
- 2 [Edit the `ImapProxyAService.cfg` file and uncomment the relevant SSL settings.](#)

- 3 If you want SSL and POP, edit the `PopProxyAService.cfg` file and uncomment the relevant SSL settings.

Additionally, you must edit the `AService.cfg` file and add `|995` after the `110` in the `ServiceList` setting.

- 4 Make sure that the `BindDN` and `BindPass` options are set in the `ImapProxyAService.cfg` and `PopProxyAService.cfg` files.

You should also set the `DefaultDomain` option to your default domain (the domain to use for unqualified user names).

If you just want server-side SSL support, you are finished. Start the MMP with the following command in the `msg-svr-base/sbin` directory:

```
start-msg mmp
```

- 5 If you do not want to use SSL between the MMP and the backend server, then set the `SSLBacksidePort` option to 0 in the `ImapProxyAService.cfg` or `PopProxyAService.cfg` MMP configuration files.

▼ To Configure MMP with Client Certificate-based Login

If you want client certificate based login, do the following:

- 1 Get a copy of a client certificate and the CA certificate which signed it.
- 2 Import the CA certificate as a Trusted Certificate Authority (see “[23.5.1 Obtaining Certificates on page 722](#)”).
- 3 Use the Store Administrator you created during your Messaging Server installation.

For more information, see the “[20.4 Specifying Administrator Access to the Store](#)” on page 594

- 4 Create a `certmap.conf` file for the MMP. For example:

```
certmap default default
default:DNCmps
default:FilterComps e=mail
```

This means to search for a match with the `e` field in the certificate DN by looking at the `mail` attribute in the LDAP server.

- 5 Edit your `ImapProxyAService.cfg` file and do the following:

- a. Set `CertMapFile` to `certmap.conf`
- b. Set `StoreAdmin` and `StorePass` to values from Step 3.

- c. Set UserGroupDN to the root of your Users and Groups tree.
- 6 If you want client certificates with POP3, repeat [Step 5](#) for the PopProxyAService.cfg file.
- 7 If the MMP is not already running, start it with the following command in the *msg-svr-base/sbin* directory:
`start-msg mmp`
- 8 Import the client certificate into your client. In Netscape™ Communicator, click on the padlock (Security) icon, then select Yours under Certificates, then select Import a Certificate... and follow the instructions.

Note – All your users will have to perform this step if you want to use client certificates everywhere.

7.4.1 A Sample Topology

The fictional Siroe Corporation has two Messaging Multiplexors on separate machines, each supporting several Messaging Servers. POP and IMAP user mailboxes are split across the Messaging Server machines, with each server dedicated exclusively to POP or exclusively to IMAP (You can restrict client access to POP services alone by removing the ImapProxyAService entry from the ServiceList setting; likewise, you can restrict client access to IMAP services alone by removing the PopProxyAService entry from the ServiceList setting.). Each Messaging Multiplexor also supports only POP or only IMAP. The LDAP directory service is on a separate, dedicated machine.

This topology is illustrated below in [Figure 7–2](#).

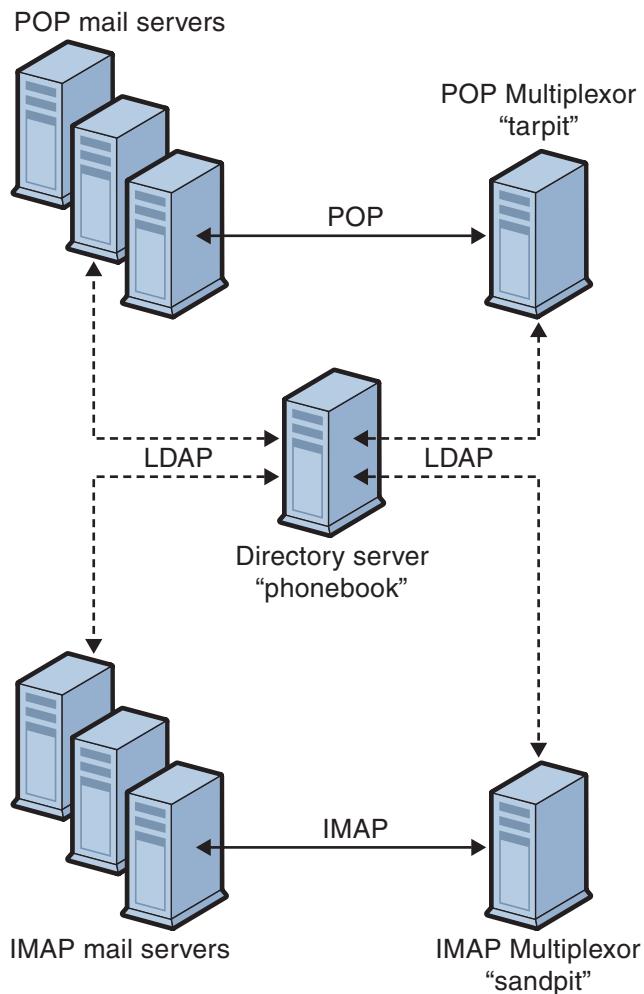


FIGURE 7–2 Multiple MMPs Supporting Multiple Messaging Servers

7.4.1.1 IMAP Configuration Example

The IMAP Messaging Multiplexor in Figure 7–2 is installed on sandpit, a machine with two processors. This Messaging Multiplexor is listening to the standard port for IMAP connections (143). Messaging Multiplexor communicates with the LDAP server on the host phonebook for user mailbox information, and it routes the connection to the appropriate IMAP server. It overrides the IMAP capability string, provides a virtual domain file, and supports SSL communications.

This is its `ImapProxyAService.cfg` configuration file:

```
default:LdapUrl ldap://phonebook.siroe.com/o=internet
default:LogDir /opt/SUNWmsgsr/config/log
default:LogLevel 5
default:BindDN "cn=Directory Manager"
default:BindPass secret
default:BacksidePort 143
default:Timeout 1800
default:Capability "IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE
UIDPLUS CHILDREN BINARY LANGUAGE XSENDER X-NETSCAPE XSERVERINFO"
default:SearchFormat (uid=%s)
default:SSLEnable yes
default:SSLPorts 993
default:SSLSecmodFile /opt/SUNWmsgsr/config/secmod.db
default:SSLCertFile /opt/SUNWmsgsr/config/cert8.db
default:SSLKeyFile /opt/SUNWmsgsr/config/key3.db
default:SSLKeyPasswdFile /opt/SUNWmsgsr/config/sslpassword.conf
default:SSLCipherSpecs all
default:SSLCertNicknames Siroe.com Server-Cert
default:SSLCacheDir /opt/SUNWmsgsr/config
default:SSLBacksidePort 993
default:VirtualDomainFile /opt/SUNWmsgsr/config/vdmap.cfg
default:VirtualDomainDelim @
default:ServerDownAlert "your IMAP server appears to be temporarily
out of service"
default:MailHostAttrs mailHost
default:PreAuth no
default:CRAMs no
default:AuthCacheSize 10000
default:AuthCacheTTL 900
default:AuthService no
default:AuthServiceTTL 0
default:BGMax 10000
default:BGPenalty 2
default:BGMaxBadness 60
default:BGDecay 900
default:BGLinear no
default:BGExcluded /opt/SUNWmsgsr/config/bgexcl.cfg
default:ConnLimits 0.0.0.0|0.0.0.0:20
default:LdapCacheSize 10000
default:LdapCacheTTL 900
default:HostedDomains yes
default:DefaultDomain Siroe.com
```

7.4.1.2

POP Configuration Example

The POP Messaging Multiplexor example in “[7.4.1 A Sample Topology](#)” on page 183 is installed on tarpit, a machine with four processors. This Messaging Multiplexor is listening to the

standard port for POP connections (110). Messaging Multiplexor communicates with the LDAP server on the host phonebook for user mailbox information, and it routes the connection to the appropriate POP server.

This is its `PopProxyAService.cfg` configuration file:

```
default:LdapUrl ldap://phonebook.siroe.com/o=internet
default:LogDir /opt/SUNWmsgsr/config/log
default:LogLevel 5
default:BindDN "cn=Directory Manager"
default:BindPass password
default:BacksidePort 110
default:Timeout 1800
default:SearchFormat (uid=%s)
default:SSLEnable no
default:VirtualDomainFile /opt/SUNWmsgsr/config/vdmap.cfg
default:VirtualDomainDelim @
default:MailHostAttrs mailHost
default:PreAuth no
default:CRAMs no
default:AuthCacheSize 10000
default:AuthCacheTTL 900
default:AuthService no
default:AuthServiceTTL 0
default:BGMax 10000
default:BGPenalty 2
default:BGMaxBadness 60
default:BGDecay 900
default:BGLinear no
default:BGExcluded /opt/SUNWmsgsr/config/bgexcl.cfg
default:ConnLimits 0.0.0.0|0.0.0.0:20
default:LdapCacheSize 10000
default:LdapCacheTTL 900
default:HostedDomains yes
default:DefaultDomain Siroe.com
```

7.5 MMP Tasks

This section describes miscellaneous MMP configuration tasks. These include:

- “[7.5.1 To Configure Mail Access with MMP](#)” on page 187
- “[7.5.2 To Set a Failover MMP LDAP Server](#)” on page 187

7.5.1

To Configure Mail Access with MMP

The MMP does not make use of the PORT_ACCESS mapping table. If you wish to reject SMTP connections from certain IP addresses and you are using the MMP, you must use the TCPAccess option. The syntax of this option is the same as mailDomainAllowedServiceAccess (see *Sun Java Communications Suite 5 Schema Reference*). It is also described at “[23.7.2 Filter Syntax](#)” on page 739.

7.5.2

To Set a Failover MMP LDAP Server

It is possible to specify more than one LDAP server for the MMP so that if one fails another takes over. Modify your PopProxyAService.cfg or ImapProxyAService.cfg to the following:

```
default:LdapUrl "ldap://ldap01.yourdomain ldap02.yourdomain/o=internet"
```

Note – Make sure there is a space between the host names in the above configuration.

MTA Concepts

This chapter provides a conceptual description of the MTA. It consists of the following sections:

- “[8.1 The MTA Functionality](#)” on page 189
- “[8.2 MTA Architecture and Message Flow Overview](#)” on page 193
- “[8.3 The Dispatcher](#)” on page 195
- “[8.4 Rewrite Rules](#)” on page 196
- “[8.5 Channels](#)” on page 197
- “[8.6 The MTA Directory Information](#)” on page 201
- “[8.7 The Job Controller](#)” on page 202

8.1 The MTA Functionality

The Message Transfer Agent, or *MTA* is a component of the Messaging Server ([Figure 8–1](#)). At its most basic level, the MTA is a message router. It accepts messages from other servers, reads the address, and routes it to the next server on way to its final destination, typically a user’s mailbox.

Over the years, a lot of functionality has been added to the MTA, and with it, size, power, and complexity. These MTA functions overlap, but, in general, can be classified as follows:

- **Routing.** Accepts a message, expands or transforms it if necessary (for example if it is an alias), and routes it to the next server, channel, program, file, or whatever. The routing function has been expanded to allow administrator specification of the internal and external mechanics of how messages are routed. For example, it is possible to specify things such as SMTP authentication, use of various SMTP commands and protocol, TCP/IP or DNS lookup support, job submission, process control and message queueing and so on.
- **Address Rewriting.** Envelope addresses are often rewritten as part of the routing process, but envelope or header addresses can also be rewritten to a more desired or appropriate form.

- **Filtering.** The MTA can filter messages based on address, domain, possible virus or spam content, size, IP address, header content, and so on. Filtered messages can be discarded, rejected, modified, sent to a file, sent to a program, or be sent to the next server on its way to a user mailbox.
- **Content Modification.** Message headers or content can be modified. Example: making a message readable to a specific client or in a specific character set or checking for spam or viruses.
- **Auditing.** Tracking who submitted what, where and when.

A number of subcomponents and processes support these functions and are shown in [Figure 8–2](#). This chapter describes these subcomponents and processes. In addition, a number of tools allow system administrators to enable and configure these functions. These include MTA options, configutil parameters, mapping tables, keywords, channels, and rewrite rules. These are described in later MTA chapters:

- [Chapter 10, “About MTA Services and Configuration”](#)
- [Chapter 11, “Configuring Rewrite Rules”](#)
- [Chapter 12, “Configuring Channel Definitions”](#)
- [Chapter 13, “Using Predefined Channels”](#)
- [Chapter 14, “Integrating Spam and Virus Filtering Programs Into Messaging Server”](#)
- [Chapter 16, “LMTP Delivery”](#)
- [Chapter 17, “Vacation Automatic Message Reply”](#)
- [Chapter 18, “Mail Filtering and Access Control”](#)
- [Chapter 23, “Configuring Security and Access Control”](#)
- [Chapter 25, “Managing Logging”](#)
- [Chapter 26, “Troubleshooting the MTA”](#)
- [Chapter 27, “Monitoring Messaging Server”](#)

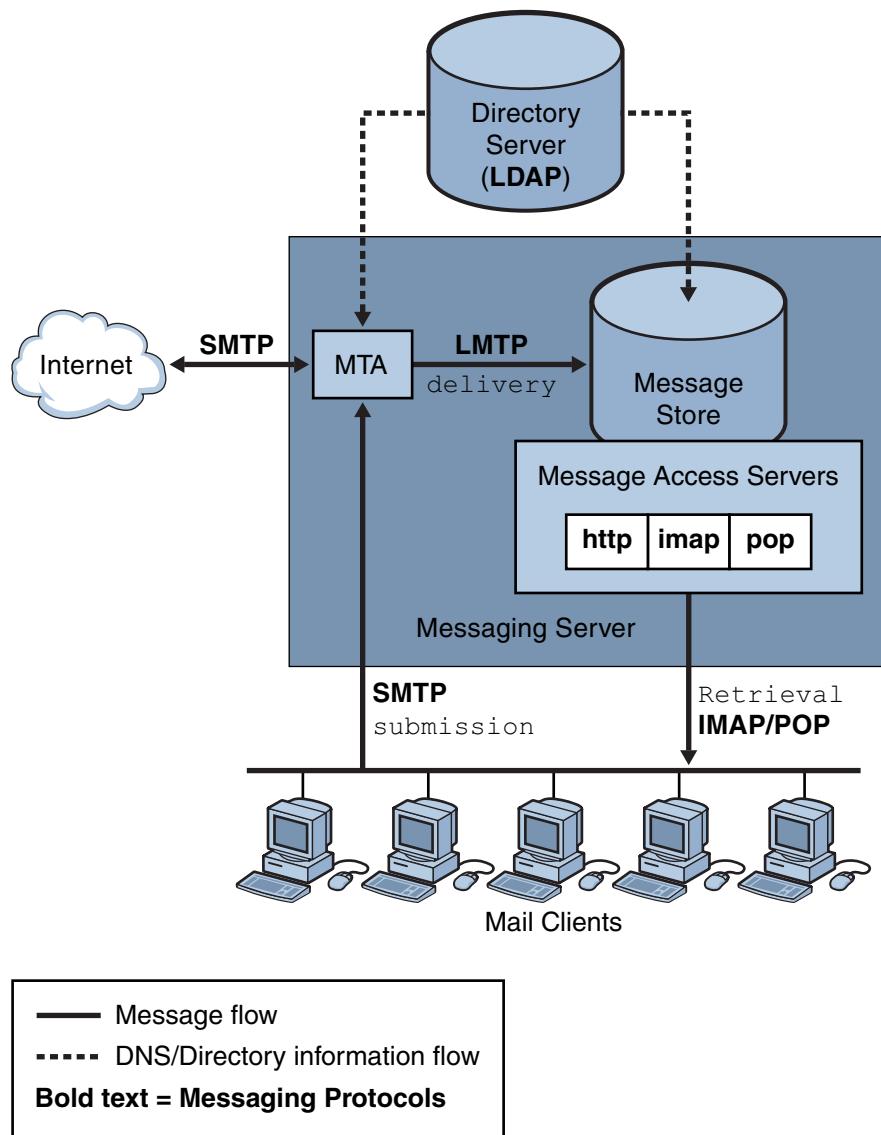


FIGURE 8–1 Messaging Server, Simplified Components View (Communications Express not Shown)

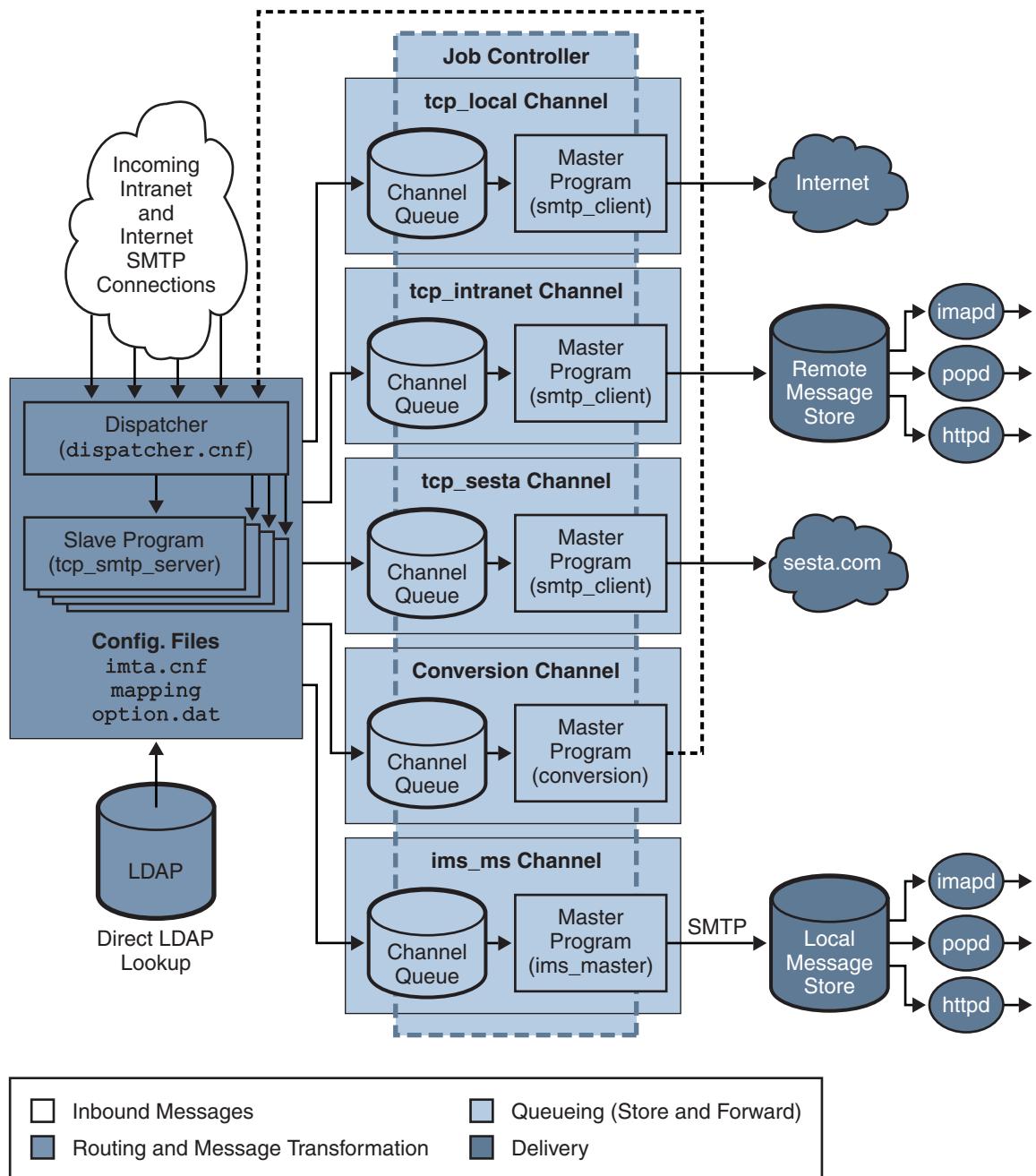


FIGURE 8–2 MTA Architecture

8.2 MTA Architecture and Message Flow Overview

This section provides a short overview of MTA architecture and message flow (Figure 8–2). Note that the MTA is a highly complex component and this figure is a *simplified* depiction of messages flowing through the system. In fact, this picture is not a perfectly accurate depiction of all messages flowing through the system. For purposes of conceptual discussion, however, it must suffice.

8.2.1

Dispatcher and SMTP Server (Slave Program)

Messages enter the MTA from the internet or intranet via SMTP sessions. When the MTA receives a request for an SMTP connection, the MTA *dispatcher* (a multithreaded connection dispatching agent), executes a *slave* program (`tcp_smtp_server`) to handle the SMTP session. The dispatcher maintains pools of multithreaded processes for each service. As additional sessions are requested, the dispatcher activates an SMTP server program to handle each session. A process in the Dispatcher's process pool may concurrently handle many connections. Together the dispatcher and slave program perform a number of different functions on each incoming message. Three primary functions are:

- Message blocking - messages from specified IP addresses, mail addresses, ports, channels, header strings and so on, may be blocked (Chapter 18, “Mail Filtering and Access Control”).
- Address changing. Incoming `From:` or `To:` addresses may be rewritten to a different form.
- Channel enqueueing. Addresses are run through the rewrite rules to determine which channel the message should be sent.

For more information see “8.3 The Dispatcher” on page 195

8.2.1.1

Routing and Address Rewriting

SMTP servers enqueue messages, but so can a number of other channels including, the conversion channel and reprocess channel. A number of tasks are achieved during this phase of delivery, but the primary tasks are:

- Alias expansion.
- Running the addresses through the rewrite rules which do two things:
 - Rewrite the domain part of addresses into a desired format.
 - Direct messages to the appropriate channel queue.

Channels

The channel is the fundamental MTA component used for message processing. A channel represents a message connection with another system (for example, another MTA, another channel, or the local message store). As mail comes in, different messages require different

routing and processing depending on the message's source and destination. For example, mail to be delivered to a local message store will be processed differently from mail to be delivered to the internet which will be processed differently from mail to be sent to another MTA within the mail system. Channels provide the mechanism for customizing the processing and routing required for each connection. In a default installation, the majority of messages go to a channels handling internet, intranet, and local messages.

Specialized channels for specific situations can also be created. For example, suppose that a certain internet domain processes mail very slowly causing mail addressed to this domain to clog up the MTA. A special channel could be created to provide special handling for messages addressed to the slow domain, thus relieving the system of this domain bottleneck.

The domain part of the address determines to what channel the message is enqueued. The mechanism for reading the domain and determining the appropriate channel is called the rewrite rules (see “[8.4 Rewrite Rules](#) on page 196”).

Channels typically consist of a channel queue and a channel processing program called a *master program*. After the slave program delivers the message to the appropriate channel queue, the master program performs the desired processing and routing. Channels, like rewrite rules, are specified and configured in the `imta.cnf` file. An example of a channel entry is shown below:

```
tcp_intranet smtp mx single_sys subdirs 20 noreverse maxjobs 7 SMTP_POOL  
maytlsserver allowswitchchannel saslswitchchannel tcp_auth  
tcp_intranet-daemon
```

The first word, in this case `tcp_intranet` is the channel name. The last word is called the channel tag. The words in between are called channel keywords and specify how messages are to be processed. Hundreds of different keywords allow messages to be processed in many ways. A complete description of channel keywords is provided in [Chapter 12, “Configuring Channel Definitions.”](#)

Message Delivery

After the message is processed, the master program sends the message to the next stop along the message's delivery path. This may be the intended recipient's mailbox, another MTA, or even a different channel. Forwarding to another channel is not shown in the picture, but is common occurrence.

Note that the local parts of addresses and received fields are typically 7-bit characters. If the MTA reads 8-bit characters in these fields, it replaces each 8-bit character with asterisks.

8.3 The Dispatcher

The Dispatcher is a multithreaded dispatching agent that permits multiple multithreaded server processes to share responsibility for SMTP connection services. When using the Dispatcher, it is possible to have several multithreaded SMTP server processes running concurrently, all handling connections to the same port. In addition, each server may have one or more active connections.

The Dispatcher acts as a central receiver for the TCP ports listed in its configuration. For each defined service, the Dispatcher may create one or more SMTP server processes to handle the connections after they are established.

In general, when the Dispatcher receives a connection for a defined TCP port, it checks its pool of available worker processes for the service on that port and chooses the best candidate for the new connection. If no suitable candidate is available and the configuration permits it, the Dispatcher may create a new worker process to handle this and subsequent connections. The Dispatcher may also create a new worker process in expectation of future incoming connections. There are several configuration options which may be used to tune the Dispatcher's control of its various services, and in particular, to control the number of worker processes and the number of connections each worker process handles.

See “[10.4.4 Dispatcher Configuration File](#)” on page 253 for more information.

8.3.1

Creation and Expiration of Server Processes

Automatic housekeeping facilities within the Dispatcher control the creation of new and expiration of old or idle server processes. The basic options that control the Dispatcher's behavior are `MIN_PROCS` and `MAX_PROCS`. `MIN_PROCS` provides a guaranteed level of service by having a number of server processes ready and waiting for incoming connections. `MAX_PROCS`, on the other hand, sets an upper limit on how many server processes may be concurrently active for the given service.

It is possible that a currently running server process might not be able to accept any connections because it is already handling the maximum number of connections of which it is capable, or because the process has been scheduled for termination. The Dispatcher may create additional processes to assist with future connections.

The `MIN_CONNS` and `MAX_CONNS` options provide a mechanism to help you distribute the connections among your server processes. `MIN_CONNS` specifies the number of connections that flags a server process as “busy enough” while `MAX_CONNS` specifies the “busiest” that a server process can be.

In general, the Dispatcher creates a new server process when the current number of server processes is less than `MIN_PROCS` or when all existing server processes are “busy enough” (the number of currently active connections each has is at least `MIN_CONNS`).

If a server process is killed unexpectedly, for example, by the UNIX system `kill` command, the Dispatcher still creates new server processes as new connections come in.

For information about configuring the Dispatcher, see “[10.4.4 Dispatcher Configuration File](#)” on page 253

8.3.2

To Start and Stop the Dispatcher

To start the Dispatcher, execute the command:

```
start-msg dispatcher
```

This command subsumes and makes obsolete any other `start-msg` command that was used previously to start up a component of the MTA that the Dispatcher has been configured to manage. Specifically, you should no longer use `imsimta start smtp`. An attempt to execute any of the obsoleted commands causes the MTA to issue a warning.

To shut down the Dispatcher, execute the command:

```
stop-msg dispatcher
```

What happens with the server processes when the Dispatcher is shut down depends upon the underlying TCP/IP package. If you modify your MTA configuration or options that apply to the Dispatcher, you must restart the Dispatcher so that the new configuration or options take effect.

To restart the Dispatcher, execute the command:

```
imsimta restart dispatcher
```

Restarting the Dispatcher has the effect of shutting down the currently running Dispatcher, then immediately starting a new one.

8.4 Rewrite Rules

Rewrite rules determine the following:

- How to rewrite the domain part of an address into its proper or desired format.
- To which channel the message should be enqueued after the address is rewritten.

Each rewrite rule consists of a *pattern* and a *template*. The pattern is a string to match against the domain part of an address. The template specifies the actions to take if the domain part matches the pattern. It consists of two things: 1) a set of instructions (that is, a string of control characters) specifying how the address should be rewritten and 2) the name of the channel to which the message shall be sent. After the address is rewritten, the message is enqueued to the destination channel for delivery to the intended recipient.

An example of a rewrite rule is shown below:

```
siroe.com $U%$D@tcp_siroe-daemon
```

siroe.com is the domain pattern. Any message with the address containing siroe.com will be rewritten as per the template instructions (\$U%\$D). \$U specifies that the rewritten address use the same user name. % specifies that the rewritten address use the same domain separator. \$D specifies that the rewritten address use the same domain name that was matched in the pattern. @tcp_siroe-daemon specifies that the message with its rewritten address be sent to the channel called tcp_siroe-daemon. See [Chapter 11, “Configuring Rewrite Rules”](#), for more details.

For more information about configuring rewrite rules, see [“10.2 The MTA Configuration File” on page 235](#) and [Chapter 11, “Configuring Rewrite Rules”](#)

8.5 Channels

The channel is the fundamental MTA component that processes a message. A channel represents a connection with another computer system or group of systems. The actual hardware connection or software transport or both may vary widely from one channel to the next.

Channels perform the following functions:

- Transmit messages to remote systems, deleting them from their queue after they are sent.
- Accept messages from remote systems, placing them in the appropriate channel queues.
- Deliver messages to the local message store.
- Deliver messages to programs for special processing.

Messages are enqueued by channels on the way into the MTA and dequeued on the way out. Typically, a message enters via one channel and leaves by another. A channel might dequeue a message, process the message, or enqueue the message to another MTA channel.

This section consists of the following subsections:

- [“8.5.1 Master and Slave Programs” on page 197](#)
- [“8.5.2 Channel Message Queues” on page 199](#)
- [“8.5.3 Channel Definitions” on page 200](#)

8.5.1

Master and Slave Programs

Generally (but not always), a channel is associated with two programs: master and slave. The slave program accepts messages from another system and adds them to a channel's message queue. The master program transfers messages from the channel to another system.

For example, an SMTP channel has a master program that transmits messages and a slave program that receives messages. These are, respectively, the SMTP client and server.

The master channel program is typically responsible for outgoing connections where the MTA has initiated the operation. The master channel program:

- Runs in response to a local request for processing.
- Dequeues the message from the channel message queue.
- If the destination format is not the same format as the queued message, performs conversion of addresses, headers, and content, as necessary.
- Initiates network transport of the message.

The slave channel program typically accepts incoming connections where the MTA is responding to an external request. The slave channel program:

- Runs in response to an external event or upon local demand.
- Enqueues a message to a channel. The target channel is determined by passing envelope addresses through a rewrite rule.

For example, [Figure 8–3](#) shows two channel programs, Channel 1 and Channel 2. The slave program in Channel 1 receives a message from a remote system. It looks at the address, applies rewrite rules as necessary, then based on the rewritten address enqueues the message to the appropriate channel message queue.

The master program dequeues the message from the queue and initiates network transport of the message. Note that the master program can only dequeue messages from its own channel queue.

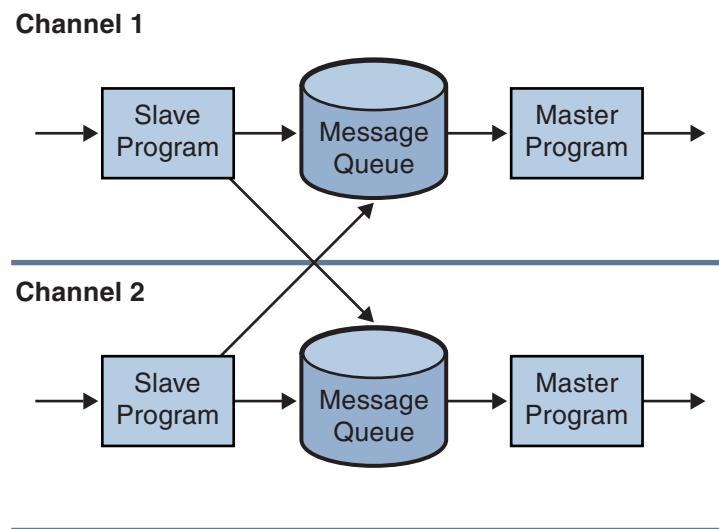


FIGURE 8–3 Master and Slave Programs

Although a typical channel has both a master and a slave program, it is possible for a channel to contain only a slave program *or* a master program. For example, the `ims-ms` channel supplied with Messaging Server contains only a master program because this channel is responsible only for dequeuing messages to the local message store, as shown in Figure 8–4.

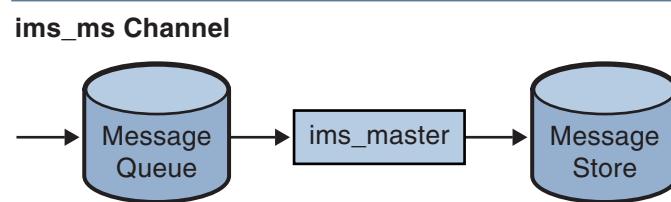


FIGURE 8–4 ims-ms Channel

8.5.2

Channel Message Queues

All channels have an associated message queue. When a message enters the messaging system, a slave program determines to which message queue the message is enqueued. The enqueued messages are stored in message files in the channel queue directories. By default, these directories are stored at the following location: `msg-svr-base/data/queue/channel/*`.

Information on message queue sizing can be found at “[Disk Sizing for MTA Message Queues](#)” in *Sun Java Communications Suite 5 Deployment Planning Guide*



Caution – Do not add any files or directories in the MTA queue directory (that is, the value of IMTA_QUEUE in the imta_tailor file) as this will cause problems. When using a separate file system for the MTA queue directories, create a subdirectory under that mount point and specify that subdirectory as the value of IMTA_QUEUE.

8.5.3

Channel Definitions

Channel definitions appear in the lower half of the MTA configuration file, imta.cnf, following the rewrite rules (see “[10.2 The MTA Configuration File](#)” on page 235 rules section and the start of the channel definitions.

A channel definition contains the name of the channel followed by an optional list of keywords that define the configuration of the channel, and a unique channel tag, which is used in rewrite rules to route messages to the channel. Channel definitions are separated by single blank lines. Comments, but no blank lines, may appear inside a channel definition.

```
[blank line]
! sample channel definition
Channel_Name keyword1 keyword2
Channel_Tag
[blank line]
```

Collectively, the channel definitions are referred to as the channel host table. An individual channel definition is called a channel block. For example, in the example below, the channel host table contains three channel definitions or blocks.

```
! test.cnf - An example configuration file.
!
! Rewrite Rules
.

.

!

! BEGIN CHANNEL DEFINITIONS
! FIRST CHANNEL BLOCK
l
local-host

! SECOND CHANNEL BLOCK
a_channel defragment charset7 usascii
a-daemon
```

```
! THIRD CHANNEL BLOCK
b_channel noreverse notices 1 2 3
b-daemon
```

A typical channel entry will look something like this:

```
tcp_intranet smtp mx single_sys subdirs 20 noreverse maxjobs 7 SMTP_POOL
maytlsserver allowswitchchannel saslswitchchannel tcp_auth
tcp_intranet-daemon
```

The first word, in this case `tcp_intranet`, is the channel name. The last word, in this case `tcp_intranet-daemon`, is called the *channel tag*. The channel tag is the name used by rewrite rules to direct messages. The words in between the channel name and channel tag are called channel *keywords* and specify how the message is to be processed. Hundreds of different keywords allow messages to be processed in many ways. A complete listing of channel keywords is listed and described in [Chapter 12, “Configuring Channel Definitions”](#)

The channel host table defines the channels Messaging Server can use and the names of the systems associated with each channel.

On UNIX systems, the first channel block in the file always describes the local channel, `l`. (An exception is a `defaults` channel, which can appear before the local channel.) The local channel is used to make routing decisions and for sending mail sent by UNIX mail tools.

You can also set global options for channels in the MTA Option file, `option.dat`, or set options for a specific channel in a channel option file. For more information on the option files, see [“10.4.6 Option File” on page 255](#), and [“10.4.2 TCP/IP \(SMTP\) Channel Option Files” on page 253](#). For details on configuring channels, see [Chapter 12, “Configuring Channel Definitions.”](#) For more information about creating MTA channels, see [“10.2 The MTA Configuration File” on page 235](#).

8.6 The MTA Directory Information

For each message that it processes, the MTA needs to access directory information about the users, groups, and domains that it supports. This information is stored in an LDAP directory service. The MTA directly accesses the LDAP directory. This is fully described in [Chapter 9, “MTA Address Translation and Routing”](#)

8.7 The Job Controller

Each time a message is enqueued to a channel, the Job Controller ensures that there is a job running to deliver the message. This might involve starting a new job process, adding a thread, or simply noting that a job is already running. If a job cannot be started because the job limit for the channel or pool has been reached, the Job Controller waits until another job has exited. When the job limit is no longer exceeded, the Job Controller starts another job.

Channel jobs run inside processing pools within the Job Controller. A pool can be thought of a “place” where the channel jobs are run. The pool provides a computing area where a set of jobs can operate without vying for resources with jobs outside of the pool. For more information on pools see “[10.4.8 Job Controller File](#)” on page 256 and “[12.5.4 Processing Pools for Channel Execution Jobs](#)” on page 384.

Job limits for the channel are determined by the `maxjobs` channel keyword. Job limits for the pool are determined by the `JOB_LIMIT` option for the pool.

Messaging Server normally attempts to deliver all messages immediately. If a message cannot be delivered on the first attempt, however, the message is delayed for a period of time determined by the appropriate `backoff` keyword. As soon as the time specified in the `backoff` keyword has elapsed, the delayed message is available for delivery, and if necessary, a channel job is started to process the message.

The Job Controller’s in-memory data structure of messages currently being processed and awaiting processing typically reflects the full set of message files stored on disk in the MTA queue area. However, if a backlog of message files on disk builds up enough to exceed the Job Controller’s in-memory data structure size limit, then the Job Controller tracks in memory only a subset of the total number of messages files on disk. The Job Controller processes only those messages it is tracking in memory. After a sufficient number of messages have been delivered to free enough in-memory storage, the Job Controller automatically refreshes its in-memory store by scanning the MTA queue area to update its list of messages. The Job Controller then begins processing the additional message files it just retrieved from disk. The Job Controller performs these scans of the MTA queue area automatically.

Previously, the Job Controller read all the files in the queue directory in the order in which they are found. It now reads several channel queue directories at once. This makes for much more reasonable behavior on startup, restart, and after `max_messages` has been exceeded. The number of directories to be read at once is controlled by the Job Controller option `Rebuild_Parallel_Channel`. This can take any value between 1 and 100. The default is 12.

If your site routinely experiences heavy message backlogs, you might want to tune the Job Controller by using the `MAX_MESSAGES` option. By increasing the `MAX_MESSAGES` option value to allow Job Controller to use more memory, you can reduce the number of occasions when message backlogs overflow the Job Controller’s in-memory cache. This reduces the overhead involved when the Job Controller must scan the MTA queue directory. Keep in mind, however, that when the Job Controller does need to rebuild the in-memory cache, the process will take

longer because the cache is larger. Note also that because the Job Controller must scan the MTA queue directory every time it is started or restarted, large message backlogs mean that starts or restarts of the Job Controller will incur more overhead than starts or restarts when no such backlog exists.

You do not want to overwhelm the job controller by keeping information about huge numbers of messages in memory. For this reason, there has to be a and upper and lower limit. The number specified by `MAX_MESSAGES` is the number of messages that the job controller will hold in memory. It will get this high if there are new messages delivered, for instance ones received by `tcp_smtp_server`. Beyond this number, messages are queued (put on disk), but not put into the job controller memory structure. The job controller notices this condition and when the number of messages in memory drops below half this maximum, it starts scanning the disk queues for more messages. It always looks for untried messages "ZZ..." files first, then previously tried messages.

In addition, the job controller limits the number of messages reclaimed from disk. It only reads from disk up to three-quarters of the `MAX_MESSAGES` to allow for headroom for new messages (if messages are being reclaimed from disk, they have been delayed, which is an undesirable state).

Furthermore, you want to avoid cluttering up the memory structure with delayed messages—those that can not be processed yet. When a message is delayed because it cannot be delivered immediately (a delivery attempt has failed if the number of messages the job controller knows about is greater than 5/8 of `MAX_MESSAGES` and the number of delayed messages is greater than 3/8 of `MAX_MESSAGES`) the message is forgotten until the next sweep of the on disk structures, which will be when the number of messages drops below 1/2 `MAX_MESSAGES`.

The only obvious problems with having `MAX_MESSAGES` too small is that the scheduling of jobs will become suboptimal. The scanning of the disk queues is also a bit simplistic. If you have huge numbers of messages backlogged in both the `tcp_local` and `ims_ms` queues, then the rebuild thread will find all the messages for one channel first, then the ones for the next channel. This can result in alarmed administrators reporting that they've fixed one issue, but are only seeing only one specific channel dequeuing.

This is not a problem. There is a memory cost of approximately 140 bytes for each message. Having a message limit of 100000, you are limiting the job controller data structures to about 20 Megabytes (there are other data structures representing jobs, channels, destination hosts and so on). This is insignificant on a big server.

All the named objects in the job controller are tracked in a hash table. This is sized at the next power of 2 bigger than `MAX_MESSAGES`, and is never re-sized. Each entry in this hash table is a pointer, so we are looking at a memory usage of four times `MAX_MESSAGES` rounded up to a power of two. Being a hash table, this will tend all to be in memory as the hash function is supposed to be random. This is another 0.5 Megabytes in the default case.

For information about pools and configuring the Job Controller, see "[10.4.8 Job Controller File](#)" on page 256 and "[12.5 Configuring Message Processing and Delivery](#)" on page 380.

8.7.1

To Start and Stop the Job Controller

To start the Job Controller, execute the command:

```
start-msg job_controller
```

To shut down the Job Controller, execute the command:

```
stop-msg job_controller
```

To restart the Job Controller, execute the command:

```
imsimta restart job_controller
```

Restarting the Job Controller has the effect of shutting down the currently running Job Controller, then immediately starting a new one.

MTA Address Translation and Routing

Prior to Messaging Server 6 2003Q4, Messaging Server used to access all user, domain, and group data from a database that was compiled from information stored in an LDAP server. When directory information was updated in the LDAP server, the database information was synchronized with a program called `dirsync`. The Messaging Server MTA now accesses the LDAP directory directly. This chapter describes the flow of data through the MTA using direct LDAP data access. It contains the following sections:

- “[9.1 The Direct LDAP Algorithm and Implementation](#)” on page 205
- “[9.2 Address Reversal](#)” on page 228
- “[9.3 Asynchronous LDAP Operations](#)” on page 230
- “[9.4 Settings Summary](#)” on page 231
- “[9.5 Processing Multiple Different LDAP Attributes with the Same Semantics](#)” on page 232

9.1 The Direct LDAP Algorithm and Implementation

The following sections describe direct LDAP processing:

- “[9.1.1 Domain Locality Determination](#)” on page 205
- “[9.1.2 Alias expansion of local addresses](#)” on page 209
- “[9.1.3 Processing the LDAP Result](#)” on page 214
- “[9.1.4 To Modify Group Membership Attribute Syntax](#)” on page 228

9.1.1 Domain Locality Determination

Starting with an address of the form *user@domain*, the address translation and routing process first checks to see if *domain* is local. This section consists of the following subsections:

- “[9.1.1.1 Rewrite Rule Machinery](#)” on page 206
- “[9.1.1.2 Domain Map Determination of Domain Locality](#)” on page 207
- “[9.1.1.3 Caching Of Domain Locality Information](#)” on page 208

- “9.1.1.4 Error Handling” on page 208
- “9.1.1.5 Pattern for Domain Check Rewrite Rule” on page 208
- “9.1.1.6 Putting It All Together” on page 209

9.1.1.1

Rewrite Rule Machinery

A facility has been added to the MTA rewrite rule machinery to check a given string to see if it is a domain we need to handle locally. This new facility is activated by a \$V or \$Z metacharacter. These new metacharacters are syntactically similar to the existing \$N, \$M, \$Q, and \$C metacharacters, that is, they are followed by a pattern string. In the case of \$N, \$M, \$Q, and \$C the pattern is matched against either the source or destination channel. In the case of \$V and \$Z the pattern is a domain and the check is to see whether or not it is local. \$V causes a rule failure for a nonlocal domain and \$Z causes a rule failure for a local domain.

The handling of these metacharacters is implemented as the following procedure:

1. Messaging Server checks to see if the current domain matches a valid domain entry in the directory. Go to step 3 if no entry exists.
2. If the domain has an entry in the directory, the attribute specified by the LDAP_DOMAIN_ATTR_ROUTING_HOSTS MTA option (default `mailRoutingHosts`) is retrieved from the domain entry. If this attribute is present, it lists the set of hosts able to handle users in this domain. This list is compared against the host specified by the `local.hostname` configutil parameter and the list of hosts specified by the `local.imta.hostnamealiases` configutil parameter. These options can be overridden by the LDAP_LOCAL_HOST and LDAP_HOST_ALIAS_LIST MTA options, respectively. If there is a match or the attribute is not present on the domain, the domain is local. If no match occurs, the domain is nonlocal.

The handling of domains considered to be nonlocal because of the `mailRoutingHosts` attribute depends on the setting of the ROUTE_TO_ROUTING_HOST MTA option. If the option is set to 0 (the default), the address is simply treated as nonlocal and MTA rewrite rules are used to determine routing. If the option is set to 1, a source route consisting of the first value listed in the LDAP_DOMAIN_ATTR_ROUTING_HOSTS MTA option is prepended to the address.

3. If no domain entry can be found, remove a component from the left hand side of the domain and go to Step 1. If no components remain continue to Step 4.

A consequence of this backtracking up the domain tree is that if `siroe.com` is recognized as local, any subdomain of `siroe.com` will be recognized as local. Situations may arise where this is undesirable, so an MTA option, DOMAIN_UPLEVEL, is provided to control this behavior. Specifically, bit 0 (value = 1) of DOMAIN_UPLEVEL, if clear, disables retries with domain components removed. The default value of DOMAIN_UPLEVEL is 0.

4. Vanity domain checking now needs to be performed. Vanity domains do not have domain entries, rather, they are specified by attaching special domain attributes to one or more user entries. The vanity domain check is done by instituting an LDAP search using the LDAP URL specified by the DOMAIN_MATCH_URL MTA option. The value of this option should be set to:

```
ldap:///B?msgVanityDomain?sub?(msgVanityDomain=$D)
```

`$B` substitutes the value of the `local.uldapbasedn configutil` parameter; this is the base of the user tree in the directory. The `LDAP_USER_ROOT` MTA option can be used to override the value of this `configutil` option specifically for the MTA.

The actual return value from this search does not matter. What matters is if there is a value to return. If there is a return value the domain is considered to be local, if not it is considered to be nonlocal.

9.1.1.2

Domain Map Determination of Domain Locality

It is also instructive to note what steps are performed to find valid domain entries in the directory. The steps are schema-level specific. In the case of Sun LDAP Schema 1, they are:

1. Convert the domain to a base DN in the domain tree. This is done by converting the domain into a series of `dc` components and then adding a domain root suffix. The default suffix is obtained from the `service.dcroot configutil` parameter. The default suffix is `o=internet`. So a domain of the form `a.b.c.d` would typically be converted into `dc=a,dc=b,dc=c,dc=d,o=internet`. The `service.dcroot configutil` parameter can be overridden by setting the `LDAP_DOMAIN_ROOT` MTA option.
2. Look for an entry with the base DN found in Step 1 and an object class of either `inetDomain` or `inetDomainAlias`. The search filter used for this purpose can be overridden by setting the `LDAP_DOMAIN_FILTER_SCHEMA1` MTA option which defaults to `(|(objectclass=inetDomain)(objectclass=inetdomainalias))`.
3. Exit with a failure if nothing is found.
4. If the object class of the entry found is `inetDomain`, check to make sure the entry has an `inetDomainBaseDn` attribute associated with the domain entry. If it is present, it is saved for use in subsequent searches for user entries and processing terminates. If it is not present, the entry is assumed to be a domain alias and processing continues with step 5. The MTA option `LDAP_DOMAIN_ATTR_BASEDN` can be used to override the use of `inetDomainBaseDN`.
5. The entry must be a domain alias; look up the new entry referenced by the `aliasedObjectName` attribute and return to step 4. Processing terminates with a failure if the `no aliasedObjectName` attribute is present. An alternative to the use of `aliasedObjectName` attribute can be specified with the MTA option `LDAP_DOMAIN_ATTR_ALIAS`.

Note that processing can return to step 4 at most once; domain aliases pointing at domain aliases are not allowed.

In Sun LDAP Schema 2, the action taken is much simpler: The directory is searched for an entry with the object class `sunManagedOrganization` where the domain appears as a value of either the `sunPreferredDomain` or `associatedDomain` attribute. If need be, the use of the `sunPreferredDomain` and `associatedDomain` attributes for this purpose can be overridden with the respective MTA options `LDAP_ATTR_DOMAIN1_SCHEMA2` and `LDAP_ATTR_DOMAIN2_SCHEMA2`. The search is done under the root specified by the `service.dcroot configutil` parameter. The `service.dcroot configutil` parameter can be

overridden by setting the `LDAP_DOMAIN_ROOT` MTA option. Additionally, domain entries in Schema 2 are not required to have `inetDomainBaseDn` attributes; if they do not, the base of the user tree is assumed to be the domain entry itself.

Two MTA options support more efficient domain lookups from user base domain names. They are `LDAP_BASEDN_FILTER_SCHEMA1`, which is a string specifying a filter used to identify Schema 1 domains when performing user base domain name searches. The default is the value of `LDAP_DOMAIN_FILTER_SCHEMA1` if that MTA option is specified. If neither option is specified the default is `(objectclass=inetDomain)`. `LDAP_BASEDN_FILTER_SCHEMA2` is a string specifying additional filter elements used to identify Schema 2 domains when performing user base domain name searches. The default is the value of `LDAP_DOMAIN_FILTER_SCHEMA2`, if that MTA option is specified. If neither option is specified, the default is an empty string.

9.1.1.3

Caching Of Domain Locality Information

Due to the frequency with which domain rewrite operations are performed and the expense of the directory queries (especially the vanity domain check) both negative and positive indications about domains need to be cached. This is implemented with an in-memory open-chained dynamically-expanded hash table. The maximum size of the cache is set by the `DOMAIN_MATCH_CACHE_SIZE` MTA option (default 100000) and the timeout for entries in the cache is set by the `DOMAIN_MATCH_CACHE_TIMEOUT` MTA option (default 600 seconds).

9.1.1.4

Error Handling

Temporary server failures during this process have to be handled carefully, since when they occur, it is impossible to know whether or not a given domain is local. Basically two outcomes are possible in such a case:

1. Return a temporary (4xx) error to the client telling it to try the address again later.
2. Accept the address but queue it to the reprocessing channel so it can be retried locally later.

Neither of these options is appropriate in all cases. For example, outcome 1 is appropriate when talking to a remote SMTP relay. But outcome 2 is appropriate when dealing with an SMTP submission from a local user.

While it would be possible in theory to handle temporary failures by using multiple rules with the same pattern, the overhead of repeating such queries, even with a cache in place, is unacceptable. For these reasons, the simple success/fall-through-to-the-next-rule matching model of domain rewriting is inadequate. Instead, a special template, specified by the MTA option `DOMAIN_FAILURE`, is used in the event of a domain lookup failure. When a `$V` operation fails, this template replaces the remainder of the current rewrite rule template being processed.

9.1.1.5

Pattern for Domain Check Rewrite Rule

This domain check needs to be performed before other rewrite rules have a chance to operate. This ordering is insured by using the special `$*` on the left hand side in the rule. The `$*` pattern is checked prior to any other rules.

9.1.1.6

Putting It All Together

Taking all the machinery described so far into account, the new rewrite rule we need in the `imta.cnf` is:

```
$*      $E$F$U%$H$V$H@localhost
```

and the value of the `DOMAIN_FAILURE` MTA option in the `option.dat` file needs to be:

```
reprocess-daemon$Mtcp_local$1M$1~-error$4000000?Temporary lookup failure
```

In this rewrite rule, `localhost` is the host name associated with the local channel. The value of the `DOMAIN_FAILURE` option shown here is the default value so it does not need to appear in `option.dat` under normal circumstances.

The ordering here is especially tricky. The MTA checks `$V` after the address is rebuilt but before the route is added. This lets the MTA to change the route in the event of a temporary lookup failure. Pending channel match checks are applied any time the insertion point changes, so the `@` after the second `$H` invokes the check. If the check succeeds, the remainder of the template applies and rewrite processing concludes. If the check fails, the rewrite fails and rewriting continues with the next applicable rewrite rule. If the check cannot be performed due to a temporary failure, template processing continues with the value specified by the `DOMAIN_FAILURE` MTA option. The value of this template first sets the routing host to `reprocess-daemon`. Then the template checks to see whether or not the MTA is dealing with a reprocessing channel of some sort or `tcp_local`. If the MTA is dealing with such a channel, the rule continues, making the routing host illegal and specifying a temporary failure as the outcome. If the MTA is not dealing with such a channel, the rule is truncated and successfully terminated, thereby rewriting the address to the `reprocess` channel.

9.1.2

Alias expansion of local addresses

Once an address has been determined to be associated with the local channel it automatically undergoes alias expansion. The alias expansion process examines a number of sources of information, including:

1. The alias file (part of the compiled configuration).
2. The alias database.
3. Alias URLs.

The exact alias sources that are checked and the order in which they are checked depends on the setting of the `ALIAS_MAGIC` MTA option in the `option.dat` file. For direct LDAP, set the option to 8764. This means that the URL specified by the `ALIAS_URL0` MTA option is checked first, then the URL specified by the `ALIAS_URL1` MTA option, then the URL specific by the `ALIAS_URL2` MTA option, and finally, the alias file. The alias database is not checked when this setting is active.

The following sections further describe alias expansion:

- “9.1.2.1 Alias Checking with LDAP URLs” on page 210
- “9.1.2.2 The \$V Metacharacter” on page 210
- “9.1.2.3 Calling a Mapping from a URL” on page 211
- “9.1.2.4 The \$R Metacharacter” on page 212
- “9.1.2.5 Determining the Attributes to Fetch” on page 213
- “9.1.2.6 Handling LDAP Errors” on page 213
- “9.1.2.7 Sanity Checks on the LDAP Result” on page 213
- “9.1.2.8 Support for Vanity Domains” on page 213
- “9.1.2.9 Support for Catchall Addresses” on page 214

9.1.2.1

Alias Checking with LDAP URLs

Checking of aliases in LDAP is implemented by specifying two special LDAP URLs as alias URLs. The first of these handles regular users and groups; vanity domains are handled by subsequent alias URLs. The first URL is specified as ALIAS_URL0:

```
ALIAS_URL0=ldap:///${V}/*?sub?${R}
```

9.1.2.2

The \$V Metacharacter

Metacharacter expansion occurs prior to URL lookup. The two metacharacters used in the ALIAS_URL0 value are \$V and \$R.

The \$V metacharacter converts the domain part of the address into a base DN. This is similar to the initial steps performed by the \$V rewrite rule metacharacter described previously in the section entitled “9.1.1.1 Rewrite Rule Machinery” on page 206. \$V processing consists of the following steps:

1. Get the base DN for user entries in the current domain.
2. Get the canonical domain associated with the current domain. In Sun LDAP Schema 1, the canonical domain name is given by the `inetCanonicalDomainName` attribute of the domain entry if the attribute is present. If the attribute is absent the canonical name is the name constructed in the obvious way from the DN of the actual domain entry. This will differ from the current domain when the current domain is an alias. The name attribute used to store the canonical name may be overridden with the `LDAP_DOMAIN_ATTR_CANONICAL MTA` option in the `option.dat` file.

In Sun LDAP Schema 2, the canonical name is simply the value of the `SunPreferredDomain` attribute.

A utility for verifying canonical domain settings for domains with overlapping user entries is provided. See “`imsimta test -domain`” in *Sun Java System Messaging Server 6.3 Administration Reference*.

3. If the base DN exists, substitute it into the URL in place of the \$V.

4. Any applicable hosted domain for this entry is now determined. This is done by comparing either the canonical domain (if bit 2 (value = 4) of DOMAIN_UPLEVEL is clear) or the current domain (if bit 2 (value = 4) of DOMAIN_UPLEVEL is set) with the `service.defaultdomain configutil` parameter. If they do not match, the entry is a member of a hosted domain. The `service.defaultdomain configutil` parameter can be overridden by setting the `LDAP_DEFAULT_DOMAIN MTA` option in the `option.dat` file.
5. If the base DN determination fails, remove a component from the left hand side of the domain and go to Step 1. The substitution fails if no components remain.

`$V` also accepts an optional numeric argument. If it is set to 1 (for example, `$1V`), a failure to resolve the domain in the domain tree will be ignored and the base of the user tree specified by the `local.ugldapbasedn configutil` option will be returned.

If the attempt to retrieve the domain's base DN succeeds, the MTA also retrieves several useful domain attributes which will be needed later. The names of the attributes retrieved are set by the following MTA options in the `option.dat` file:

- `LDAP_DOMAIN_ATTR_UID_SEPARATOR` (default `domainUidSeparator`)
- `LDAP_DOMAIN_ATTR_SMARTHOST` (default `mailRoutingSmartHost`)
- `LDAP_DOMAIN_ATTR_CATCHALL_ADDRESS` (default `mailDomainCatchallAddress`)
- `LDAP_DOMAIN_ATTR_CATCHALL_MAPPING` (no default value)
- `LDAP_DOMAIN_ATTR_BLOCKLIMIT` (default `mailDomainMsgMaxBlocks`)
- `LDAP_DOMAIN_ATTR_REPORT_ADDRESS` (default `mailDomainReportAddress`)
- `LDAP_DOMAIN_ATTR_STATUS` (default `inetDomainStatus`)
- `LDAP_DOMAIN_ATTR_MAIL_STATUS` (default `mailDomainStatus`)
- `LDAP_DOMAIN_ATTR_CONVERSION_TAG` (default `mailDomainConversionTag`)
- `LDAP_DOMAIN_ATTR_FILTER` (default `mailDomainSieveRuleSource`)
- `LDAP_DOMAIN_ATTR_DISK_QUOTA` (no default)
- `LDAP_DOMAIN_ATTR_MESSAGE_QUOTA` (no default)
- `LDAP_DOMAIN_ATTR_AUTOREPLY_TIMEOUT` (no default)
- `LDAP_DOMAIN_ATTR_NOSOLICIT` (no default)
- `LDAP_DOMAIN_ATTR_OPTIN` (no default)
- `LDAP_DOMAIN_ATTR_RECIPIENTLIMIT` (no default)
- `LDAP_DOMAIN_ATTR_RECIPIENTCUTOFF` (no default)
- `LDAP_DOMAIN_ATTR_SOURCEBLOCKLIMIT` (no default)

9.1.2.3

Calling a Mapping from a URL

There might be unusual cases where the mapping from domain to base DN is done some other way. In order to accommodate such setups, the URL resolution process has the ability to call an MTA mapping. This is done with a metacharacter sequence of the general form:

`$|/mapping-name/mapping-argument|`

The double quotation ("") initiates and terminates the callout. The character immediately following the \$ is the separator between the mapping name and argument; a character should be chosen that does not collide with the expected character values used in either the mapping name or argument.

9.1.2.4

The \$R Metacharacter

The \$R metacharacter provides an appropriate filter for the URL. The intent is to produce a filter that searches all the attributes that might contain an email address for a particular user or group. The list of attributes to search comes from the configutil parameter `local.imta.mailaliases`. If this parameter is not set, the `local.imta.schematag` configutil parameter is examined, and depending on its value, an appropriate set of default attributes is chosen as follows:

```
sims401 mail, rfc822mailalias  
nms41 mail, mailAlternateAddress  
ims50 mail, mailAlternateAddress, mailEquivalentAddress
```

The value of `local.imta.schematag` can be a comma-separated list. If more than one schema is supported, the combined list of attributes with duplicates eliminated is used. The `LDAP_SCHEMATAG` MTA option can be used to override the setting of `local.imta.schematag` specifically for the MTA.

Additionally, the filter searches not only for the address that was originally supplied, but also for an address with the same local part but the domain that was actually found in the domain tree, which was saved in the second step in the section entitled “[9.1.2.2 The \\$V Metacharacter](#)” on page 210. The iterative nature of the domain tree lookup means the two addresses may be different. This additional check is controlled by bit 1 (value = 2) of the `DOMAIN_UPLEVEL` MTA option in the `option.dat` file. Setting the bit enables the additional address check. The default value of `DOMAIN_UPLEVEL` is 0.

For example, suppose that the domain `siroe.com` appears in the domain tree. Assuming Sun LDAP Schema 1 is in force, a lookup of the address

```
u@host1.siroe.com
```

The filter that results from the expansion of \$R and an ims50 schematag looks like:

```
(| (mail=u@siroe.com)  
    (mail=u@host1.siroe.com)  
    (mailAlternateAddress=u@siroe.com)  
    (mailAlternateAddress=u@host1.siroe.com)  
    (mailEquivalentAddress=u@siroe.com)  
    (mailEquivalentAddress=u@host1.siroe.com))
```

If, on the other hand, `DOMAIN_UPLEVEL` was set to 1 rather than 3, the filter would be:

```
(|(mail=u@host1.siroe.com)
 (mailAlternateAddress=u@host1.siroe.com)
 (mailEquivalentAddress=u@host1.siroe.com))
```

9.1.2.5

Determining the Attributes to Fetch

If the URL specifies an * for the list of attributes to return, we replace the asterisk with the list of attributes the MTA is able to use. This list is dynamically generated from the various MTA option settings that specify the options the MTA consumes.

9.1.2.6

Handling LDAP Errors

At this point the resulting URL is used to perform an LDAP search. If an LDAP error of some kind occurs, processing terminates with a temporary failure indication (4xx error in SMTP). If the LDAP operation succeeds but fails to produce a result, the catchall address attribute for the domain retrieved from the `LDAP_DOMAIN_ATTR_CATCHALL_ADDRESS` MTA option is checked. If it is set, its value replaces the current address.

If no catchall address attribute is set, the smarthost attribute for the domain retrieved from the `LDAP_DOMAIN_ATTR_SMARTHOST` MTA option is checked. If it is set, an address of the form

`@smarthost:user@domain`

is created and alias processing terminates successfully with this result. Additionally, the conversion tag for the domain obtained from the `LDAP_DOMAIN_ATTR_CONVERSION_TAG` MTA option will, if present, be attached to the address so that conversions can be done prior to forwarding to the smarthost. If no catchall address or smarthost exists for the domain, processing of this alias URL terminates unsuccessfully.

9.1.2.7

Sanity Checks on the LDAP Result

After the LDAP search has returned a result, it is checked to verify that there is only one entry in it. If there are more than one, each entry is checked to see if it has the right object class for a user or a group, a non-deleted status, and for users, a UID. Entries that do not pass this check are ignored. If the list of multiple entries is reduced to one by this check, processing proceeds. If not, a duplicate or ambiguous directory error is returned.

9.1.2.8

Support for Vanity Domains

The `ALIAS_URL0` check is for a conventional user or a user in a hosted domain. If this fails a vanity domain check is also made. This is done with the following alias URL:

`ALIAS_URL1=ldap:///B?*?sub?(&(msgVanityDomain=$D)$R)`

9.1.2.9

Support for Catchall Addresses

Finally, a check for a catchall address of the form `@host` needs to be made in the `mailAlternateAddress` attribute. This form of wildcarding is allowed in both hosted and vanity domains, so the proper alias URL for it is:

```
ALIAS_URL2=ldap:///${V}*?sub?(mailAlternateAddress=@$D)
```

Note – The `++*` subaddress substitution mechanism has always worked with catch-all addresses in direct LDAP mode, but the string that was substituted was the subaddress only, not the entire local part. This has been changed so that the entire local part of the original address will be plugged into the catch-all address as a subaddress when this construct is used.

For example, given an address of the form `foo+bar@domain.com`, no local user `foo` in the `domain.com` domain, and a catch-all address for `domain.com` of `b1tch++@example.com`, the resulting address will now be `b1tch+foo+bar@example.com`. It used to be `b1tch+bar@example.com`.

9.1.3

Processing the LDAP Result

LDAP alias result processing is done in a number of order-dependent stages. These stages are described in the following sections:

- “9.1.3.1 Object Class Check” on page 214
- “9.1.3.2 Entry Status Checks” on page 215
- “9.1.3.3 UID Check” on page 217
- “9.1.3.4 Message Capture” on page 217
- “9.1.3.5 Seeding the Reversal Cache” on page 217
- “9.1.3.6 Mail Host and Routing Address” on page 218
- “9.1.3.7 Miscellaneous Attribute Support” on page 219
- “9.1.3.8 Delivery Options Processing” on page 220
- “9.1.3.9 Additional Metacharacters for Use in Delivery Options” on page 221
- “9.1.3.10 Delivery Option Defaults” on page 223
- “9.1.3.11 Start and End Date Checks” on page 223
- “9.1.3.12 Optin and Presence Attributes” on page 223
- “9.1.3.13 Sieve Filter Handling” on page 224
- “9.1.3.14 Deferred Processing Control” on page 224
- “9.1.3.15 Group Expansion Attributes” on page 224

9.1.3.1

Object Class Check

If the alias search succeeds, the object class of the entry is checked to make sure it contains an appropriate set of object classes for a user or a group. The possible sets of required object classes for users and groups is normally determined by what schemata are active. This is determined by the `local.imta.schematag` setting.

Table 9–1 shows the user and group object classes that result from various schematag values.

TABLE 9–1 Object Classes Resulting from Various schematag Values

schematag	User Object Classes	Group Object Classes
sims40	inetMailRouting+inetmailuser	inetMailRouting+inetmailgroup
nms41	mailRecipient + nsMessagingServerUser	mailGroup
ims50	inetLocalMailRecipient+ inetmailuser	inetLocalMailRecipient + inetmailgroup

The information in this table, like the rest of schema tag handling, is hard coded. However, there are also two MTA options in the option.dat file, `LDAP_USER_OBJECT_CLASSES` and `LDAP_GROUP_OBJECT_CLASSES`, which can be set to specify different sets of object classes for users and groups respectively.

For example, a schema tag setting of `ims50, nms41` would be equivalent to the following option settings:

```
LDAP_USER_OBJECT_CLASSES=inetLocalMailRecipient+inetmailuser,  
mailRecipient+nsMessagingServerUser
```

```
LDAP_GROUP_OBJECT_CLASSES=inetLocalMailRecipient+inetmailgroup, mailGroup
```

The LDAP result is simply ignored if it does not have a correct set of object classes appropriate for a user or a group. The MTA also determines if it is dealing with a user or a group and saves this information. This saved information will be used repeatedly later.

Note that the object class settings described here are also used to construct an actual LDAP search filter that can be used to check to see that an entry has the right object classes for a user or a group. This filter is accessible through the `$K` metacharacter. It is also stored internally in the MTA's configuration for use by channel programs and is written to the MTA option file, `option.dat`, as the `LDAP_UG_FILTER` option when the command `imsimta cnbuild -option` is used. This option is only written to the file. The MTA never reads it from the option file.

9.1.3.2

Entry Status Checks

Next the entry's status is checked. There are two status attributes, one for the entry in general and another specifically for mail service.

Table 9–2 shows the general and mail-specific user or group attributes in the schema tag entry to check against depending on what schema tag are in effect

TABLE 9–2 Attributes to Check Against

schematag	Type	General	Mail-specific
sims40	users	inetsubscriberstatus	mailuserstatus
sims40	groups	none	inetmailgroupstatus
nms41	users	none	mailuserstatus
nms41	groups	none	none
Messaging Server 5.0	users	inetuserstatus	mailuserstatus
Messaging Server 5.0	groups	none	inetmailgroupstatus

If necessary the `LDAP_USER_STATUS` and `LDAP_GROUP_STATUS` MTA options in the `option.dat` file can be used to select alternate general status attributes for users and groups respectively. The mail-specific user and group status attributes are controlled by the `LDAP_USER_MAIL_STATUS` and `LDAP_GROUP_MAIL_STATUS` MTA options.

Another factor that plays into this are the statuses for the domain itself (`LDAP_DOMAIN_ATTR_STATUS` and `LDAP_DOMAIN_ATTR_MAIL_STATUS`). All in all there are four status attributes. They are combined by considering them in the following order:

1. Domain status
2. Domain mail status
3. User or group status
4. Mail user or group status

The first of these that specifies something other than “active” takes precedence over all the others. The other permissible status values are “inactive,” “deleted,” “removed,” “disabled,” “hold,” and “overquota.” “Hold,” “disabled,” and “removed” statuses may only be specified for mail domains, mail users, or mail groups. “Overquota” status can only be specified as a mail domain or mail user status.

All statuses default to “active” if a particular status attribute is not present. Unknown status values are interpreted as “inactive.”

When the four statuses are combined, the following statuses for a user or group are possible: “active,” “inactive,” “deleted,” “removed,” “disabled,” “hold,” and “overquota.” Active status causes alias processing to continue. Inactive or overquota status results in immediate rejection of the address with a 4xx (temporary) error. Deleted, removed, and disabled statuses results in immediate rejection of the address with a 5xx (permanent) error. Hold status is treated as active as far as status handling is concerned but it sets an internal flag so that when delivery options are considered later on, any options that are there are overridden with an option list containing a single “hold” entry.

9.1.3.3

UID Check

The next step is to consider the entry's UID. UIDs are used for a variety of purposes and must be part of all user entries and may be included in group entries. A user entry without a UID is ignored and processing of this alias URL terminates unsuccessfully. UIDs for entries in a hosted domain can consist of the real UID, a separator character, and then a domain. The MTA only wants the real UID so the rest is removed if present using the domain separator character obtained with the `LDAP_DOMAIN_ATTR_UID_SEPARATOR` MTA option in the `option.dat` file.

In the unlikely event that an attribute other than `uid` is used to store UIDs, the `LDAP_UID` MTA option can be used to force use of a different attribute.

9.1.3.4

Message Capture

Next the LDAP attribute used to specify one or more message capture addresses is checked. The attribute used for this purpose must be specified with the `LDAP_CAPTURE` MTA option. There is no default. Values of this attribute are treated as addresses and a special “capture” notification is generated and sent to these address containing the current message as an attachment. Additionally, the capture addresses are used to seed the address reversal cache in the likely event the address will subsequently appear as an envelope from: address.

9.1.3.5

Seeding the Reversal Cache

Next the primary address and any aliases attached to the user entry are considered. This information is used to seed the address reversal cache. It plays no part in the current address translation process. First, the primary address, personal name, recipient limit, recipient cutoff, and source block limit attributes are considered. The primary address is normally stored in the “mail” attribute; another attribute can be specified by setting the `LDAP_PRIMARY_ADDRESS` MTA option appropriately. (The primary address reverses to itself, of course.) There is no default attribute for any of the other attributes. If you want to use them, you must specify them with the `LDAP_PERSONAL_NAME`, (see “[17.4 Vacation Autoreply Attributes](#)” on page 537) `LDAP_RECIPIENTLIMIT`, `LDAP_RECIPIENTCUTOFF`, (see “[12.9.7 Limiting Message Recipients](#)” on page 414) and `LDAP_SOURCEBLOCKLIMIT` (see “[12.9.2 Specifying Absolute Message Size Limits](#)” on page 411) MTA options. The corresponding domain-level recipient limit, recipient cutoff, and source block limit attributes are also considered at this point. User-level settings completely override any domain-level setting.

Next, any secondary addresses are considered and a cache entry is made for each one. There are two sorts of secondary addresses: Those that undergo address reversal and those that do not. Both must be considered in order to properly seed the address reversal cache because of the need to check for message capture requests in all cases.

Secondary addresses that undergo reversal are normally stored in the `mailAlternateAddress` attribute. Another attribute can be specified by setting the `LDAP_ALIAS_ADDRESSES` MTA option. Secondary addresses that do not undergo reversal are normally stored in the `mailEquivalentAddress` attribute. Another attribute can be specified with the `LDAP_EQUIVALENCE_ADDRESSES` MTA option.

9.1.3.6

Mail Host and Routing Address

It is now time to consider the `mailhost` and `mailRoutingAddress` attributes. The actual attributes considered can be overridden with the `LDAP_MAILHOST` and `LDAP_ROUTING_ADDRESS` MTA options, respectively. These attributes work together to determine whether or not the address should be acted on at this time or forwarded to another system.

The first step is to decide whether or not `mailhost` is meaningful for this entry. A preliminary check of the delivery options active for the entry is done to see whether or not the entry is mailhost-specific. If it is not, `mailhost` checking is omitted. See the description of “[9.1.3.8 Delivery Options Processing](#)” on page 220, and the `#` flag in particular, to understand how this check is done.

In the case of a user entry, the `mailhost` attribute must identify the local system in order to be acted on. The `mailhost` attribute is compared with the value of the `local.hostname` configutil parameter and against the list of values specified by the `local.imta.hostnamealiases` configutil parameter. The `mailhost` attribute is deemed to identify the local host if any of these match.

A successful match means that the alias can be acted on locally and alias processing continues. An unsuccessful match means that the message needs to be forward to the mailhost to be acted on. A new address of the form

@mailhost:user@domain

is constructed and becomes the result of the alias expansion operation.

The handling of a missing `mailhost` attribute is different depending on whether the entry is a user or a group. In the case of a user, a mailhost is essential, so if no `mailhost` attribute is present a new address of the form

@smarthost:user@domain

is constructed using the smart host for the domain determined by the `LDAP_DOMAIN_ATTR_SMARTHOST` MTA option. An error is reported if no smart host exists for the domain.

Groups, on the other hand, do not require a mailhost, so a missing mailhost is interpreted as meaning that the group can be expanded anywhere. So alias processing continues.

The `mailRoutingAddress` attribute adds one final wrinkle. Its presence causes processing to terminate with the `mailRoutingAddress` as the result. In version 5.2, the `mailHost` check was done first and must pass before the routing address can take effect. To get the same behavior in the current version of Messaging Server, the format of the `mailRoutingAddress` attribute could be as follows: `mailRoutingAddress: @mailhost:user@domain`

9.1.3.7

Miscellaneous Attribute Support

Next the `mailMsgMaxBlocks` attribute is considered. First it is minimized with the domain block limit returned from the `LDAP_DOMAIN_ATTR_BLOCKLIMIT` MTA option. If the size of the current message is known to exceed the limit, alias processing terminates with a size-exceeded error. If the size is not known or does not exceed the limit, the limit is nevertheless stored and will be rechecked when the message itself is checked later. The use of `mailMsgMaxBlocks` can be overridden with the `LDAP_BLOCKLIMIT` MTA option.

Next a number of attributes are accessed and saved. Eventually these will be written into the queue file entry for use by the `ims_master` channel program, which will then use them to update the store's user information cache. If the attributes are not found for individual users, domain-level attributes can be used to set defaults.

This step is skipped if the LDAP entry is for a group rather than a user or if the LDAP entry came from the alias cache and not from the LDAP directory. The logic behind the latter criteria is that frequent updates of this information are unnecessary and using the alias cache offers a reasonable criteria for when updates should be done. The names of the attributes retrieved are set by various MTA options.

Table 9–3 shows the MTA options which set the retrieved disk quota and message quota attributes.

TABLE 9–3 MTA Options Which Set the Retrieved Disk Quota and Message Quota Attributes

MTA option	Attribute
<code>LDAP_DISK_QUOTA</code>	<code>mailQuota</code>
<code>LDAP_MESSAGE_QUOTA</code>	<code>mailMsgQuota</code>

Next a number of attributes are stored for possible use in conjunction with metacharacter substitutions later.

Table 9–4 shows the MTA options, the default attribute, and metacharacters.

TABLE 9–4 MTA Options, Default Attributes, and Metacharacters

MTA Option	Default Attribute	Metacharacters
<code>LDAP_PROGRAM_INFO</code>	<code>mailProgramDeliveryInfo</code>	<code>\$P</code>
<code>LDAP_DELIVERY_FILE</code>	<code>mailDeliveryFileURL</code>	<code>\$F</code>
<code>LDAP_SPARE_1</code>	no default	<code>\$1E \$1G \$E</code>
<code>LDAP_SPARE_2</code>	no default	<code>\$2E \$2G \$G</code>

TABLE 9-4 MTA Options, Default Attributes, and Metacharacters (Continued)

MTA Option	Default Attribute	Metacharacters
LDAP_SPARE_3	no default	\$3E \$3G
LDAP_SPARE_4	no default	\$4E \$4G
LDAP_SPARE_5	no default	\$5E \$5G

Spare slots for additional attributes are included so that you can use them to build customized address expansion facilities.

Next any values associated with the `mailconversiontag` attribute are added to the current set of conversion tags. The name of this attribute can be changed with the `LDAP_CONVERSION_TAG` MTA option. If any values were associated with the domain's `mailDomainConversionTag` attribute, they are attached as well.

9.1.3.8 Delivery Options Processing

Next the `mailDeliveryOption` attribute is checked. The name of this attribute can be changed with the `LDAP_DELIVERY_OPTION` MTA option. This is a multi valued option and its values determine the addresses produced by the alias translation process. Additionally, the permissible values are different for users and groups. Common permissible values are `program`, `forward`, and `hold`. User-only values are `mailbox`, `native`, `unix`, and `autoreply`. The group-only values are `members`, `members_offline`, and `file`.

The conversion of the `mailDeliveryOption` attribute into appropriate addresses is controlled by the `DELIVERY_OPTIONS` MTA option. This option not only specifies what addresses are produced by each permissible `mailDeliveryOption` value, but also what the permissible `mailDeliveryOption` values are and whether or not each one is applicable to users, groups, or both.

The value of this option consists of a comma-separated list of `deliveryoption=template` pairs, each pair with one or more optional single character prefixes.

The default value of the `DELIVERY_OPTIONS` option is:

```
DELIVERY_OPTIONS=*mailbox=$M%$\\$2I$_+$2S@ims-ms-daemon, \
&members=*, \
*native=$M@native-daemon, \
/hold=@hold-daemon:$A, \
*unix=$M@native-daemon, \
&file=+$F@native-daemon, \
&@members_offline=*, \
program=$M%$P@pipe-daemon, \
#forward=**, \
*^!autoreply=$M+$D@bitbucket
```

Each delivery option corresponds to a possible `mailDeliveryOption` attribute value and the corresponding template specifies the resulting address using the same metacharacter substitution scheme used by URL processing.

Table 9–5 shows the single character prefixes available for the `DELIVERY_OPTIONS` options.

TABLE 9–5 Single-Character Prefixes for options in the `DELIVERY_OPTIONS` MTA option

Character Prefix	Description
@	Sets a flag saying that the message needs to be redirected to the reprocess channel. Processing of the current user/group is abandoned. Flag ignored for messages originating from the reprocess channel.
*	Delivery option applies to users.
&	Delivery option applies to groups.
\$	Sets a flag saying expansion of this user or group is to be deferred.
^	Sets a flag saying that the vacation start and end times should be checked to see if this delivery option really is in effect.
#	Sets a flag saying expansion of this delivery option does not need to take place on the entry's designated mailhost. That is, the following entry is mailhost-independent. This lets the MTA check to see if all of a given user or group's delivery options are independent of the mailhost. If this condition is satisfied the MTA can act on the entry immediately rather than having to forward the message to the mailhost.
/	Sets a flag that causes all addresses produced by this delivery option to be held. Message files containing these recipient addresses will have a <code>.HELD</code> extension.
!	Sets a flag that says that autoreply operations should be handled internally by the MTA. It only makes sense to use this prefix on an autoreply delivery option. The option's value should direct the message to the bitbucket channel

If neither * nor & are present, the delivery option is taken to apply to both users and groups.

9.1.3.9

Additional Metacharacters for Use in Delivery Options

Several additional metacharacters have been added to support this new use of the MTA's URL template facility. These include:

Table 9–6 shows additional metacharacters and their descriptions for use in delivery options.

TABLE 9–6 Additional Metacharacters for Use in Delivery Options

Metacharacter	Description
<code>\$\</code>	Force subsequent text to lower case.
<code>\$^</code>	Force subsequent text to upper case.
<code>\$_</code>	Perform no case conversion on subsequent text.

TABLE 9–6 Additional Metacharacters for Use in Delivery Options *(Continued)*

Metacharacter	Description
\$nA	Insert the <i>n</i> th character of the address. The first character is character 0. The entire address is substituted if <i>n</i> is omitted. This is intended to be used to construct autoreply directory paths.
\$D	Insert the domain part of the address.
\$nE	Insert the value of the <i>n</i> th spare attribute. If <i>n</i> is omitted the first attribute is used.
\$F	Insert the name of the delivery file (<code>mailDeliveryFileURL</code> attribute).
\$nG	Insert the value of the <i>n</i> th spare attribute. If <i>n</i> is omitted the second attribute is used.
\$nH	Insert the <i>n</i> th component of the domain from the original address counting from 0. The default is 0 if <i>n</i> is omitted.
\$nI	Insert hosted domain associated with alias. This metacharacter accepts an integer parameter <i>n</i> whose semantics are described in Table 9–7 .
\$nJ	Insert the <i>n</i> th part of the host domain counting from 0. The default for <i>n</i> is 0.
\$nO	Insert source route associated with the current address. This metacharacter accepts an integer parameter <i>n</i> whose semantics are described in Table 9–7 .
\$K	Insert an LDAP filter that matches the object classes for a user or group. See the description of the <code>LDAP_UG_FILTER</code> output-only MTA option.
\$L	Insert the local part of the address.
\$nM	Insert the <i>n</i> th character of the UID. The first character is character 0. The entire UID is substituted if <i>n</i> is omitted.
\$P	Insert the program name (<code>mailProgramDeliveryInfo</code> attribute).
\$nS	Insert subaddress associated with the current address. This metacharacter accepts an integer parameter <i>n</i> whose semantics are described in Table 9–7 .
\$nU	Insert the <i>n</i> th character of the dequoted form of the mailbox part of the current address. The first character is character 0. The entire dequoted mailbox is substituted if <i>n</i> is omitted.
\$nX	Insert the <i>n</i> th component of the mailhost. The entire mailhost is inserted if <i>n</i> is omitted.

[Table 9–7](#) shows how the integer parameter modifies the behavior of the \$nI and \$nS metacharacters.

TABLE 9–7 Integers Controlling Behavior Modification of the \$nI and \$nS Metacharacters

Integer	Description of Behavior
0	Fail if no value is available (default).
1	Insert value if one is available. Insert nothing if not.
2	Insert value if one is available. Insert nothing and delete preceding character if one is not (this particular behavior is needed by the ims-ms channel).

TABLE 9-7 Integers Controlling Behavior Modification of the \$nI and \$nS Metacharacters
(Continued)

Integer	Description of Behavior
3	Insert value if one is available. Insert nothing and ignore following character if one is not.

In addition to the metacharacters, [Table 9-8](#) shows two special template strings.

TABLE 9-8 Special Template Strings

Special Template String	Description
*	Perform group expansion. This value is not valid for user entries.
**	Expand the attribute named by the LDAP_FORWARDING_ADDRESS MTA option. This defaults to <code>mailForwardingAddress</code> .

With group expansion, for example, if a user's `mailDeliveryOption` value is set to `mailbox`, we form a new address consisting of the stripped UID, a percent sign followed by the hosted domain if one is applicable, a plus sign followed by the subaddress if one was specified, and finally `@ims-ms-daemon`.

9.1.3.10

Delivery Option Defaults

If the list of active delivery options is empty at this point, the first option on the list (usually `mailbox`) is activated for users and the second option on the list (usually `members`) is activated for groups.

9.1.3.11

Start and End Date Checks

Start and end dates are checked after the delivery option list has been read. There are two attributes whose names are controlled by the `LDAP_START_DATE` (default `vacationStartDate`) and `LDAP_END_DATE` (default `vacationEndDate`) MTA options, respectively. If one or more of the active delivery options specified the ^ prefix character, the values of these options are checked against the current date. If the current date is outside the range specified by these options, the delivery options with the ^ prefix are removed from the active set. For more information see [“17.4 Vacation Autoreply Attributes” on page 537](#).

9.1.3.12

Optin and Presence Attributes

The `LDAP_OPTIN1` through `LDAP_OPTIN8` MTA options specify LDAP attributes for per-user spam filter opt-in values based on destination addresses. If an option is specified and the attribute is present, it is appended to the current spam filter opt-in list. Any values set by the domain level attribute set by the `LDAP_DOMAIN_ATTR_OPTIN` MTA option will also be appended to the list. `LDAP_SOURCE_OPTIN1` through `LDAP_SOURCE_OPTIN8` provide comparable originator-address-based per-user spam filter optins.

The LDAP_PRESENCE MTA option can be used to specify a URL that can be resolved to return presence information about the user. If the option is specified and the attribute is present, its value is saved for possible use in conjunction with Sieve presence tests. The domain level attribute set by the LDAP_DOMAIN_ATTR_PRESENCE MTA option is used as source for this URL if no value exists for the user entry.

9.1.3.13 Sieve Filter Handling

Next the mailSieveRuleSource attribute is checked for a Sieve filter that applies to this entry. If this attribute exists, it is parsed and stored at this point. The two possible forms for the value of this attribute are a single value that contains a complete Sieve script or multiple values where each value contains a piece of a Sieve script. The latter form is produced by the web filter construction interface. Special code is used to order the values and glue them together properly.

The use of the mailSieveRuleSource attribute specifically can be overridden by using the LDAP_FILTER MTA option.

9.1.3.14 Deferred Processing Control

Next the mailDeferProcessing attribute is checked. This attribute can be changed by using the LDAP_REPROCESS MTA option. Processing continues normally if this attribute exists and is set to no. But if this attribute is set to yes and the current source channel is not the reprocess channel, expansion of this entry is aborted and the original *user@domain* address is simply queued to the reprocess channel. If this attribute does not exist, the setting of the deferred processing character prefix associated with delivery options processing is checked. (See the section “[9.1.3.8 Delivery Options Processing](#)” on page 220 the default for users is no. The default for groups is controlled by the MTA option DEFER_GROUP_PROCESSING, which defaults to 1 (yes). Alias processing concludes at this point for user entries.

9.1.3.15 Group Expansion Attributes

A number of additional attributes are associated with group expansion and must be dealt with at this point. The names of these attributes are all configurable via various MTA options.

[Table 9–9](#) lists the default attribute names, the MTA option to set the attribute name, and the way the attribute is processed by the MTA. The ordering of the elements in the table shows the order in which the various group attributes are processed. This ordering is essential for correct operation.

TABLE 9–9 Group Expansion Default Attributes and MTA Option to Set

Default Attribute	(MTA option to Set Attribute Name) How the Attribute is Processed
mgrpMsgRejectAction	(LDAP_REJECT_ACTION) Single valued attribute that controls what happens if any of the subsequent access checks fail. Only one value is defined: TOMODERATOR, which if set instructs the MTA to redirect any access failures to the moderator specified by the mgrpModerator attribute. The default (and any other value of this attribute) causes an error to be reported and the message rejected.
mailRejectText	(LDAP_REJECT_TEXT) The first line of text stored in the first value of this attribute is saved. This text will be returned if any of the following authentication attributes cause the message to be rejected. This means the text can appear in SMTP responses so value has to be limited to US-ASCII to comply with current messaging standards.
mgrpBroadcasterPolicy	(LDAP_AUTH_POLICY) Specifies level of authentication needed to send to the group. Possible tokens are SMTP_AUTH_REQUIRED or AUTH_REQ, both of which mean that the SMTP AUTH command must be used to identify the sender in order to send to the group; SMTP_AUTH_USED and AUTH_USED which are similar in effect to SMTP_AUTH_REQUIRED and AUTH_REQ, but do not require posters to authenticate; PASSWORD_REQUIRED, PASSWD_REQUIRED, or PASSWD_REQ, all of which mean the password to the list specified by the mgrpAuthPassword attribute must appear in an Approved: header field in the message; OR, which changes the OR_CLAUSES MTA option setting to 1 for this list; AND, which changes the OR_CLAUSES MTA option setting to 0 for this list; and NO_REQUIREMENTS, which is non-operational. Multiple values are allowed and each value can consist of a comma-separated list of tokens. If SMTP AUTH is called for it also implies that any subsequent authorization checks will be done against the email address provided by the SASL layer rather than the MAIL FROM address.
mgrpAllowedDomain	(LDAP_AUTH_DOMAIN) Domains allowed to submit messages to this group. A match failure with the OR_CLAUSES MTA option set to 0 (the default) means access checking has failed and all subsequent tests are bypassed. A match failure with the OR_CLAUSES MTA option set to 1 sets a “failure pending” flag; some other access check must succeed in order for access checking to succeed. This check is bypassed if the submitter has already matched an LDAP_AUTH_URL. Can be multivalued and glob-style wildcards are allowed.
mgrpDisallowedDomain	(LDAP_CANT_DOMAIN) Domains not allowed to submit messages to this group. A match means access checking has failed and all subsequent checks are bypassed. This check is bypassed if the submitter has already matched an LDAP_AUTH_URL. Can be multivalued and glob-style wildcards are allowed.

TABLE 9–9 Group Expansion Default Attributes and MTA Option to Set *(Continued)*

Default Attribute	(MTA option to Set Attribute Name) How the Attribute is Processed
mgrpAllowedBroadcaster	<p>(LDAP_AUTH_URL) URL identifying mail addresses allowed to send mail to this group. Can be multivalued. Each URL is expanded into a list of addresses and each address is checked against the current envelope from: address. A match failure with the OR_CLAUSES MTA option set to 0 (the default) means access checking has failed and all subsequent tests are bypassed. A match failure with the OR_CLAUSES MTA option set to 1 sets a “failure pending” flag; some other allowed access check must succeed in order for access checking to succeed. A match also disables subsequent domain access checks. The expansion that is performed is similar to an SMTP EXPN with all access control checks disabled.</p> <p>List expansion in the context of the <code>mgrpallowedbroadcaster</code> LDAP attribute now includes all the attributes used to store email addresses (normally <code>mail</code>, <code>mailAlternateAddress</code>, and <code>mailEquivalentAddress</code>). Previously only <code>mail</code> attributes were returned, making it impossible to send to lists restricted to their own members using alternate addresses.</p>
mgrpDisallowedBroadcaster	<p>(LDAP_CANT_URL) URL identifying mail addresses not allowed to send mail to this group. Can be multivalued. Each URL is expanded into a list of addresses and each address is checked against the current envelope from: address. A match means access checking has failed and all subsequent checks are bypassed. The expansion that is performed is similar to an SMTP EXPN with all access control checks disabled.</p>
mgrpMsgMaxSize	<p>(LDAP_ATTR_MAXIMUM_MESSAGE_SIZE) Maximum message size in bytes that can be sent to the group. This attribute is obsolete but still supported for backwards compatibility; the new <code>mailMsgMaxBlocks</code> attribute should be used instead.</p>
mgrpAuthPassword	<p>(LDAP_AUTH_PASSWORD) Specifies a password needed to post to the list. The presence of a <code>mgrpAuthPassword</code> attribute forces a reprocessing pass. As the message is enqueued to the reprocessing channel, the password is taken from the header and placed in the envelope. Then, while reprocessing, the password is taken from the envelope and checked against this attribute. Additionally, only passwords that actually are used are removed from the header field.</p> <p>The OR_CLAUSES MTA option acts on this attribute in the same way it acts on the other access check attributes.</p>
mgrpModerator	<p>(LDAP_MODERATOR_URL) The list of URLs given by this attribute to be expanded into a series of addresses. The interpretation of this address list depends on the setting of the LDAP_REJECT_ACTION MTA option. If LDAP_REJECT_ACTION is set to TOMODERATOR, this attribute specifies the moderator address(es) the message is to be sent to should any of the access checks fail. If LDAP_REJECT_ACTION is missing or has any other value, the address list is compared with the envelope from address. Processing continues if there is a match. If there is no match, the message is again sent to all of the addresses specified by this attribute. Expansion of this attribute is implemented by making the value of this attribute the list of URLs for the group. Any list of RFC822 addresses or DNs associated with the group is cleared, and the delivery options for the group are set to <code>members</code>. Finally, subsequent group attributes listed in this table are ignored.</p>
mgrpDeliverTo	(LDAP_GROUP_URL1) List of URLs which, when expanded, provides a list of mailing list member addresses.
memberURL	(LDAP_GROUP_URL2) Another list of URLs which, when expanded, provides another list of mailing list member addresses.

TABLE 9–9 Group Expansion Default Attributes and MTA Option to Set *(Continued)*

Default Attribute	(MTA option to Set Attribute Name) How the Attribute is Processed
uniqueMember	(LDAP_GROUP_DN) List of DNs of group members. DNs may specify an entire subtree. Unique member DNs are expanded by embedding them in an LDAP URL. The exact URL to use is specified by the GROUP_DN_TEMPLATE MTA option. The default value for this option is: ldap:///\$A??sub?mail=* \$A specifies the point where the uniqueMember DN is inserted.
mgrpRFC822MailMember	(LDAP_GROUP_RFC822) Mail addresses of members of this list.
rfc822MailMember	(LDAP_GROUP_RFC822) rfc822MailMember is supported for backwards compatibility. Either rfc822MailMember or mgrpRFC822MailMember, but not both, can be used in any given group.
mgrpErrorsTo	(LDAP_ERRORS_TO) Sets the envelope originator (MAIL FROM) address to whatever the attribute specifies.
mgrpAddHeader	(LDAP_ADD_HEADER) Turns the headers specified in the attribute into header trimming ADD options.
mgrpRemoveHeader	(LDAP_REMOVE_HEADER) Turns the headers specified into header trimming MAXLINES=-1 options.
mgrpMsgPrefixText	(LDAP_PREFIX_TEXT) Adds the specified text to the beginning of the message text, if any.
mgrpMsgSuffixText	(LDAP_SUFFIX_TEXT) Adds the specified text to the ending of the message text, if any.
No Default	(LDAP_ADD_TAG) Checks the subject for the specified text; if it isn't present the text is added at the beginning of the subject field.

One final attribute is checked in the special case of group expansion as part of an SMTP EXPN command: mgmMemberVisibility or expandable. The LDAP_EXPANDABLE MTA option can be used to select different attributes to check. Possible values are: anyone, which means that anyone can expand the group, all or true, which mean that the user has to successfully authenticate with SASL before expansion will be allowed, and none, which means that expansion is not allowed. Unrecognized values are interpreted as none. If the attribute is not present, the EXPANDABLE_DEFAULT MTA option controls whether the expansion is allowed.

Alias entries are cached in a fashion similar to domain entries. The MTA options controlling the alias cache are ALIAS_ENTRY_CACHE_SIZE (default 1000 entries) and ALIAS_ENTRY_CACHE_TIMEOUT (default 600 seconds). The entire LDAP return value for a given alias is retained in the cache.

Negative caching of alias entries is controlled by the ALIAS_ENTRY_CACHE_NEGATIVE MTA option. A nonzero value enables caching of alias match failures. A zero value disables it. Negative caching of alias entries is disabled by default. The theory is that repeated specification of an invalid address is unlikely to occur very often in practice. In addition, negative caching may interfere with timely recognition of new users added to the directory. However, sites should consider re enabling negative caching of aliases in situations where vanity domains are heavily used. The search performed by the URL specified in ALIAS_URL0 is less likely to be successful.

9.1.4

To Modify Group Membership Attribute Syntax

Support has been added for postprocessing LDAP expansion results with a mapping. The new `LDAP_URL_RESULT_MAPPING` MTA option can be used to specify the name of a group attribute which in turn specifies the name of a mapping. This mapping will be applied to any results returned by expanding either a `grpDeliverTo` or `memberURL` attribute. The mapping probe will be of the form:

LDAP-URL|LDAP-result

If the mapping returns with `$Y` set the mapping result string will replace the LDAP result for alias processing purposes. If the mapping returns with `$N` set the result will be skipped.

This mechanism can be used to define groups based on attributes that don't contain proper email address. For example, suppose a company has placed pager numbers in all their user entries. Messages can be sent to these numbers via email by suffixing them with a particular domain. A group could then be defined as follows:

1. Define a new `grpURLResultMapping` attribute in the directory and set the `LDAP_URL_RESULT_MAPPING` MTA option to this attribute's name.
2. Define a page-all group with the following attributes:

```
grpDeliverTo: ldap:///o=usergroup?pagerTelephoneNumber?sub  
grpURLResultMapping: PAGER-NUMBER-TO-ADDRESS
```

3. Define the mapping:

```
PAGER-NUMBER-TO-ADDRESS  
*|* "$1"@pagerdomain.com$Y
```

Even more interesting effects can be achieved by combining this mechanism with the `PROCESS_SUBSTITUTION` mechanism described in “[10.12.1 Optimizing Authorization Checks to the LDAP Directory for Messages Addressed to Mailing Lists](#)” on page 285. For example, it would be easy to create a metagroup where sending to an address of the form

pager+user@domain.com

sends a page to the user named `user`.

9.2 Address Reversal

Address reversal with direct LDAP starts with a `USE_REVERSE_DATABASE` value of 4, which disables the use of any reverse database. You should also set `USE_TEXT_DATABASES` to read the `IMTA_TABLE:reverse.txt` file, as the sleepycat databases are being deprecated. It then builds on the routing facilities previously discussed. In particular, in the previous version, it began with a reverse URL specification of the form:

```
REVERSE_URL=ldap:///{$V?mail?sub?{$Q}
```

The \${V} metacharacter has already been described in the context of alias URLs. However, the \${Q} metacharacter, while quite similar in function to the \${R} metacharacter used in alias URLs, is specifically intended for use in address reversal. Unlike \${R}, it produces a filter that searches attributes containing addresses that are candidates for address reversal. The list of attributes to search comes from the MTA option `LDAP_MAIL_REVERSES`. If this option is not set, the `local.imta.schematag configutil` parameter is examined, and depending on its value, an appropriate set of default attributes is chosen.

Note – Changing `REVERSE_URL` for any reason is discouraged.

Table 9–10 shows the `local.imta.schematag` values and the default attributes chosen.

TABLE 9–10 `local.imta.schematag` Values and Attributes

Schema Tag Value	Attributes
sims40	mail, rfc822mailalias
nms41	mail, mailAlternateAddress
ims50	mail, mailAlternateAddress

The use of \${Q} is no longer appropriate, however. In order for message capture and other facilities to work correctly, address reversal has been enhanced to pay attention to the attribute that matched in addition to the fact that a match occurred. This means that \${R} should be used to specify the filter instead of \${Q}. Additionally, the \${N} metacharacter has been added, which returns a list of the attributes of interest to address reversal.

The value of \${N} cannot exactly be controlled: the MTA constructs it from its own, hard-coded (and subject to change) list of the relevant attributes for address reversal purposes. If you use the various `LDAP_*` global MTA options to change what the MTA thinks are the names of attributes of interest, you will, in fact, fetch different attributes from LDAP. But it is always whatever attributes that correspond semantically to the MTA's idea of relevant attributes. These are: `LDAP_CAPTURE` (no default), `LDAP_RECIPIENTLIMIT` (no default), `LDAP_RECIPIENTCUTOFF` (no default), `LDAP_SOURCEBLOCKLIMIT` (no default), `LDAP_SOURCE_CHANNEL` (no default), `LDAP_PERSONAL_NAME` (no default), `LDAP_SOURCE_CONVERSION_TAG` (no default), `LDAP_PRIMARY_ADDRESS` (`mail`), `LDAP_ALIAS_ADDRESSES` (`mailAlternateAddress`), `LDAP_EQUIVALENCE_ADDRESSES` (`mailEquivalentAddress`), plus the `LDAP_SPARE_*` attributes.

The resulting option value is:

```
REVERSE_URL=ldap:///{$V?${N}?sub?{$R}
```

As always, `local.imta.schematag` can be a comma-separated list. If more than one schema is supported, the combined list of attributes, with duplicates eliminated, is used.

Additionally, the filter searches not only for the address that was originally supplied, but also for an address with the same local part but the domain that was actually found in the domain tree (which was saved in step 2 of “[9.1.1.1 Rewrite Rule Machinery](#)” on page 206). The iterative nature of the domain tree lookup means the two addresses may be different.

For example, suppose that the domain `siroe.com` appears in the domain tree and the MTA looks the address:

`u@host1.siroe.com`

The filter that results from the expansion of `$R` and an `ims50` schema tag will be something like:

```
((mail=u@siroe.com)
 (mail=u@host1.siroe.com)
 (mailAlternateAddress=u@siroe.com)
 (mailAlternateAddress=u@host1.siroe.com)
 (mailEquivalentAddress=u@siroe.com)
 (mailEquivalentAddress=u@host1.siroe.com))
```

Reverse lookup returns several attributes, and the MTA knows to use the mail attribute (more precisely, the attribute named by `LDAP_PRIMARY_ADDRESS`) as the one for address reversal. Note that the `mailEquivalentAddress` (more precisely, the attribute named by `LDAP_EQUIVALENCE_ADDRESSES`) is also permitted.

After the URL is constructed an LDAP search is performed. If the search is successful, LDAP returns multiple attributes in essentially arbitrary order. Unsuccessful searches or errors leave the original address unchanged.

Due to the frequency with which address reversal operations are performed, especially given the number of addresses that can appear in a message header, and the expense of the directory queries involved, both negative and positive results need to be cached. This is implemented with an in-memory, open-chained, dynamically-expanded hash table. The maximum size of the cache is set by the `REVERSE_ADDRESS_CACHE_SIZE` MTA option (default 100000) and the timeout for entries in the cache is set by the `REVERSE_ADDRESS_CACHE_TIMEOUT` MTA option (default 600 seconds). The cache actually stores addresses themselves, not the LDAP URLs and LDAP results.

9.3 Asynchronous LDAP Operations

Asynchronous lookups avoid the need to store an entire large LDAP result in memory, which can cause performance problems in some cases. The MTA provides the ability to perform various types of lookups done by the MTA asynchronously.

Use of asynchronous LDAP lookups is controlled by the MTA option `LDAP_USE_ASYNC`. This option is a bit-encoded value. Each bit, if set, enables the use of asynchronous LDAP lookups in conjunction with a specific use of LDAP within the MTA.

Table 9–11 shows the bit and value settings for the LDAP_USE_ASYNC MTA option in the option.dat file.

TABLE 9-11 Settings for the LDAP_USE_ASYNC MTA Option

Bit	Value	Specific Use of LDAP
0	1	LDAP_GROUP_URL1 (mgrpDeliverTo) URLs
1	2	LDAP_GROUP_URL2 (memberURL) URLs
2	4	LDAP_GROUP_DN (UniqueMember) DNs
3	8	auth_list, moderator_list, sasl_auth_list, and sasl_moderator_list nonpositional list parameter URLs
4	16	cant_list, sasl_cant_list nonpositional list parameter URLs
5	32	originator_reply nonpositional list parameter URLs
6	64	deferred_list, direct_list, hold_list, nohold_list nonpositional list parameter URLs
7	128	username_auth_list, username_moderator_list, username_cant_list nonpositional list parameter URLs
8	256	alias file list URLs
9	512	alias database list URLs
10	1024	LDAP_CANT_URL (mgrpDisallowBroadcaster) outer level URLs
11	2048	LDAP_CANT_URL inner level URLs
12	4096	LDAP_AUTH_URL (mgrpAllowBroadcaster) outer level URLs
13	8192	LDAP_AUTH_URL inner level URLs
14	16384	LDAP_MODERATOR_URL (mgrpModerator) URLs

The default value of the LDAP_USE_ASYNC MTA option is 0, which means that asynchronous LDAP lookups are disabled by default.

9.4 Settings Summary

In order to enable direct LDAP, the following MTA options need to be set:

```
ALIAS_MAGIC=8764
ALIAS_URL0=ldap:///$V?*?sub?$R
USE_REVERSE_DATABASE=4
USE_DOMAIN_DATABASE=0
REVERSE_URL=ldap:///$V?mail?sub?$Q
```

If vanity domains are to be supported, the following additional options must be set:

```
DOMAIN_MATCH_URL=ldap:///${B?msgVanityDomain?sub? \\\n    (msgVanityDomain=$D)}\nALIAS_URL1=ldap:///${B?*?sub? (&(msgVanityDomain=$D)$R)}\nALIAS_URL2=ldap:///${V?*?sub?(mailAlternateAddress=@$D)}
```

Note that the last of these options also handle the case of wild carded local parts in hosted as well as vanity domains. If wild carded local part support is desired but vanity domain support is not, the following option should be used instead:

```
ALIAS_URL1=ldap:///${V?*?sub?&(mailAlternateAddress=@$D)}
```

The `filter ssrd:$A` clause needs to be removed from the `ims-ms` channel definition in the MTA configuration file (`imta.cnf`).

9.5 Processing Multiple Different LDAP Attributes with the Same Semantics

The MTA now has the ability to process multiple different LDAP attributes with the same semantics. Note that this is not the same as processing of multiple values for the same attribute, which has always been supported. The handling attributes receive depends on the semantics of the attribute. The possible options are:

1. Multiple different attributes don't make sense and render the user entry invalid. In Mail Server Version 6.2 and later this handling is the default for all attributes unless otherwise specified.
2. If multiple different attribute are specified, one is chosen at random and used.
LDAP_AUTOREPLY SUBJECT, LDAP_AUTOREPLY TEXT, and LDAP_AUTOREPLY_TEXT_INT all receive this handling in version 6.2 only; in 6.3 and later they receive the handling described in “[17.4 Vacation Autoreply Attributes](#)” on page 537. 6.3 adds the LDAP_SPARE_3 and LDAP_PERSONAL_NAME attribute to this category. Note that this was how all attributes were handled prior to 6.2.
3. Multiple different attributes do make sense and should all be acted on. This handling is currently in effect for LDAP_CAPTURE, LDAP_ALIAS_ADDRESSES, LDAP_EQUIVALENCE_ADDRESSES and LDAP_DETOURHOST_OPTIN. Note that LDAP_DETOURHOST_OPTIN attribute was first added to Version 6.3.

About MTA Services and Configuration

This chapter describes general MTA services and configuration. More specific and detailed explanations may be found in other chapters. It consists of the following sections:

- “[10.1 Compiling the MTA Configuration](#)” on page 233
- “[10.2 The MTA Configuration File](#)” on page 235
- “[10.3 Mappings File](#)” on page 237
- “[10.4 Other MTA Configuration Files](#)” on page 251
- “[10.5 Aliases](#)” on page 261
- “[10.6 Command Line Utilities](#)” on page 263
- “[10.7 SMTP Security and Access Control](#)” on page 263
- “[10.8 Log Files](#)” on page 264
- “[10.9 To Convert Addresses from an Internal Form to a Public Form](#)” on page 264
- “[10.10 Controlling Delivery Status Notification Messages](#)” on page 271
- “[10.11 Controlling Message Disposition Notifications](#)” on page 284
- “[10.12 Optimizing MTA Performance](#)” on page 285

10.1 Compiling the MTA Configuration

Whenever an MTA configuration file such as `imta.cnf`, `mappings`, `aliases`, or `option.dat` is modified, you must recompile the configuration. This compiles the configuration files into a single image in shared memory (on UNIX) or a dynamic link library (NT).

The compiled configuration has a static and dynamic reloadable part. If the dynamic part is changed, and you run an `imsimta reload`, a running program will reload the dynamic data. The dynamic parts are mapping tables, aliases, and lookup tables.

The main reason for compiling configuration information is performance. Another feature of using a compiled configuration is that configuration changes can be tested more conveniently since the configuration files themselves are not “live” when a compiled configuration is in use.

Whenever a component of the MTA (such as a channel program) must read the configuration file, it first checks to see if a compiled configuration exists. If it does, the image is attached to the running program. If the image attach operation fails, the MTA falls back on the old method of reading the text files instead.

If you make changes to the reverse, forward, or general databases, then issue the command `imsimta reload` to get the changes to take effect. If you make changes to the `imta.cnf`, `mappings` file, `aliases`, `conversions`, or `option.dat` files that do not affect the job controller, then you should issue an `imsimta cnbuild` followed by an `imsimta restart smtp`. If you make changes to `dispatcher.cnf` you need to do an `imsimta restart dispatcher`. If you make changes to the configuration files that are included in the compiled configuration that affect the job controller, but not the SMTP server, in many cases you should issue the following commands: `imsimta cnbuild` and `imsimta restart job_controller`.

If you make changes to the configuration files that are included in the compiled configuration that affect both the SMTP server and the job controller, you should issue the following commands:

```
imsimta cnbuild  
imsimta restart smtp  
imsimta restart job_controller
```

(See “[MTA Commands](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for details on these commands.)

Other instances where you must restart the job controller:

- Changes to the controller configuration files, `job_controller.cnf` or `job_controller.site`, or any files included into `job_controller.cnf`
- Adding or changing use of the channel keywords `pool`, `maxjobs`, `master`, `slave`, `single`, `single_sys`, or `multiple` in the `imta.cnf` file. Adding or changing a `threaddepth` channel keyword in `imta.cnf` can be dealt with instead via `imsimta cache -change -thread_depth=...`
- If you want changes to master channel jobs to take effect immediately (rather than waiting for the controller to time-out existing channel jobs), then any relevant (which means almost all) changes to the MTA configuration or channel option files. (Changes to the `mappings` file or to MTA databases: (1) aren't usually relevant for outbound channel jobs, though they may matter to “intermediate” channels such as `conversion`, `process`, `reprocess`, and (2) if those intermediate channels are a concern, changes to the `mappings` file or to databases can often be handled via `imsimta reload` avoiding a restart of the Job Controller.) The desire to have changes take effect immediately needs to be balanced against the damage that restarting the Job Controller will do--and also consider just how much longer a particular sort of job is going to run anyhow.

The MTA configuration includes `imta.cnf` and all files it includes (such as `internet.rules`), the `alias` file, the `mappings` file, the `conversions` file, the `option.dat` file (and any files any of the preceding include), and `imta.filter`, and the `reverse`, `forward`, and general data files, and potentially some `configutil` parameters.

Note that any changes above (such as additions/changes to keywords on channel definitions) to `imta.cnf` also require an `imsimta cnbuild`--that's a basic, regardless of whether a Job Controller restart is needed.

Try to avoid restarting the Job Controller, especially at times of large numbers of messages in the queues, unless one of the above conditions necessitates a restart.

We do not recommend use of the `imsimta refresh` command on production systems because it is often not necessary to restart the job controller and restarting the job controller will reset message retries, delayed notification messages, bounced messages, and so on.

10.2 The MTA Configuration File

The primary MTA configuration file is `imta.cnf`. By default, this file is found at `msg-svr-base/config/imta.cnf`. This file contains MTA channel definitions as well as the channel rewrite rules. The channel associated with a rewritten destination address becomes the destination channel. The system will typically work well using default `imta.cnf`.

This section provides a brief introduction to the MTA configuration file. For details about configuring the rewrite rules and channel definitions that make up the MTA configuration file, see [Chapter 11, “Configuring Rewrite Rules,”](#) and [Chapter 12, “Configuring Channel Definitions.”](#)

By modifying the MTA configuration file, you establish the channels in use at a site and establish which channels are responsible for which sorts of addresses via rewrite rules. The configuration file establishes the layout of the email system by specifying the transport methods available (channels) and the transport routes (rewrite rules) associating types of addresses with appropriate channels.

The configuration file consists of two parts: domain rewriting rules and channel definitions. The domain rewriting rules appear first in the file and are separated from the channel definitions by a blank line. The channel definitions are collectively referred to as the channel table. An individual channel definition forms a channel block.

The following example of an `imta.cnf` configuration file shows how rewrite rules are used to route messages to the proper channel. No domain names are used to keep things as simple as possible. The rewrite rules appear in the upper half of the configuration file followed by the channel definitions in the lower half of the configuration file.

```
! test.cnf - An example configuration file.  (1)!  
! This is only an example of a configuration file. It serves
```

```
! no useful purpose and should not be used in a real system.  
!  
! Part I: Rewrite rules  
a      $U@a-daemon          (2)  
b      $U@b-daemon  
c      $U%c@b-daemon  
d      $U%d@a-daemon  
      (3)  
! Part II: Channel definitions  
l      (4)  
local-host  
  
a_channel defragment charset7 usascii      (5)  
a-daemon  
  
b_channel noreverse notices 1 2 3  
b-daemon  
  
</opt/SUNWmsgsr/msg-tango/table/internet.rules    (6)
```

The key items (labeled with boldface numbers, enclosed in parentheses) in the preceding configuration file are explained in the following list:

1. Exclamation points (!) are used to include comment lines. The exclamation point must appear in the first column. An exclamation point appearing anywhere else is interpreted as a *literal* exclamation point.
2. The rewrite rules appear in the first half of the configuration file. No blank lines can appear among the lines of rewrite rules. Lines with comments (beginning with an exclamation point in the first column) are permitted.
3. The first blank line to appear in the file signifies the end of the rewrite rules section and the start of the channel blocks. These definitions are collectively referred to as the *channel host table*, which defines the channels that the MTA can use and the names associated with each channel.
4. The first channel block to appear is usually the local or l channel. Blank lines then separate each channel block from one another. (An exception is the defaults channel, which can appear before the l channel).
5. A typical channel definition consists of a channel name (a_channel) some keywords which define the configuration of a channel (defragment charset7 usascii) and a routing system (a-daemon), which is also called a *channel tag*.
6. The contents of other files may be included in the configuration file. If a line is encountered with a less than sign (<) in column one, the rest of the line is treated as a file name; the file name should always be an absolute and full file path. The file is opened and its contents are spliced into the configuration file at that point. Include files may be nested up to three levels deep. Any files included in the configuration file must be world-readable just as the configuration file is world-readable.

Table 10–1 shows how some example addresses would be routed by the preceding configuration.

TABLE 10–1 Addresses and Associated Channels

Address	Queued to channel
u@a	a_channel
u@b	b_channel
u@c	b_channel
u@d	a_channel

Refer to “8.4 Rewrite Rules” on page 196, “8.5.3 Channel Definitions” on page 200, and Chapter 11, “Configuring Rewrite Rules” for more information on the MTA configuration file.

Note – Whenever changes are made to the `imta.cnf` file, the MTA configuration must be recompiled. See “10.1 Compiling the MTA Configuration” on page 233.

10.3 Mappings File

Many components of the MTA employ table lookup-oriented information. This type of table is used to transform, that is, *map*, an input string into an output string. Such tables, called *mapping tables*, are usually presented as two columns. The first (left-hand) column provides possible input strings against which to match (pattern), and the second (right-hand) column gives the resulting output string to which the input string is mapped (template). For details about which MTA processes uses which tables and when, see Table 10–2.

Most of the MTA databases that contain different types of MTA data and which should not be confused with mapping tables—are instances of just this type of table. The MTA database files, however, do not provide wildcard-lookup facilities, owing to inherent inefficiencies in having to scan the entire database for wildcard matches.

The MTA `mappings` file supports multiple mapping tables. Wildcard capabilities are provided, as well as multistep and iterative mapping methods. This approach is more compute-intensive than using a database, especially when the number of entries is large. However, the attendant gain in flexibility may serve to eliminate the need for most of the entries in an equivalent database, and this may result in lower overhead overall.

Mapping tables are kept in the MTA `mappings` file. This is the file specified with the `IMTA_MAPPING_FILE` option in the MTA `tailor` file; by default, this is `msg-svr-base/config/mappings`. The contents of the `mappings` file is incorporated into the compiled configuration as part of the reloadable section (see “10.1 Compiling the MTA

[Configuration](#) on page 233 Failure to allow world-read access leads to erratic behavior. Whenever changes are made to the mappings file, the MTA configuration must be recompiled. See “[10.1 Compiling the MTA Configuration](#)” on page 233

Table 10–2 lists the mapping tables described in this book.

TABLE 10–2 Messaging Server Mapping Tables

Mapping Table	Page	Description
AUTH_REWRITE		Used with the authrewrite keyword to modify header and envelope addresses using addressing information obtained from authentication operations (SASL). See “ 12.4.3 TCP/IP Connection and DNS Lookup Support ” on page 368
CHARSET_CONVERSION		Used to specify what sorts of channel-to-channel character set conversions and message reformatting should be done. See “ 13.6 Character Set Conversion and Message Reformatting ” on page 452
COMMENT_STRINGS		Used to modify address header comments (strings enclosed in parentheses). See “ 12.6.13 Handling Comments in Address Header Lines ” on page 396
CONVERSATIONS		Used to select message traffic for the conversion channel. See “ 13.5.2 Selecting Traffic for Conversion Processing ” on page 434
FORWARD		Used to perform forwarding similar to that performed using the alias file or alias database. See “ 10.9.3 The Forward Lookup Table and FORWARD Address Mapping ” on page 268
FROM_ACCESS		Used to filter mail based on envelope From addresses. Use this table if the To address is irrelevant. See “ 18.2.1 Access Control Mapping Tables—Operation ” on page 542
INTERNAL_IP		Used to recognize systems and subnets that are internal. See “ 18.6 To Add SMTP Relaying ” on page 557
IP_ACCESS		Used to block incoming connections based on source channel, IP address count for remote server, index of current IP address being tried. See “ 18.3.5 IP_ACCESS Mapping Table ” on page 554
MAIL_ACCESS		Used to block incoming connections based on combined information found in SEND_ACCESS and PORT_ACCESS table. See “ 18.2.1 Access Control Mapping Tables—Operation ” on page 542
NOTIFICATION_LANGUAGE		Used to customize or localize notification messages. See “ 10.10 Controlling Delivery Status Notification Messages ” on page 271
ORIG_MAIL_ACCESS		Used to block incoming connections based on combined information found in ORIG_SEND_ACCESS and PORT_ACCESS tables. See “ 18.2.1 Access Control Mapping Tables—Operation ” on page 542

TABLE 10–2 Messaging Server Mapping Tables (Continued)

Mapping Table	Page	Description
ORIG_SEND_ACCESS		Used to block incoming connections based on envelope From address, envelope To address, source and destination channels. See “ 18.2.1 Access Control Mapping Tables—Operation ” on page 542
PERSONAL_NAMES		Used to modify personal names (strings preceding angle-bracket-delimited addresses). See “ 12.6.14 Handling Personal Names in Address Header Lines ” on page 397
PORT_ACCESS		Used to block incoming connections based on IP number. See “ 18.2.1 Access Control Mapping Tables—Operation ” on page 542
REVERSE		Used to convert addresses from an internal form to a public, advertised form. “ 10.9 To Convert Addresses from an Internal Form to a Public Form ” on page 264
SEND_ACCESS		Used to block incoming connections based on envelope From address, envelope To address, source and destination channels. See “ 18.2.1 Access Control Mapping Tables—Operation ” on page 542
SMS_Channel_TEXT		Used for site-defined text conversions. See “ C.2.5 Site-defined Text Conversions ” on page 930
X-ATT-NAMES		Used to retrieve a parameter value from a mapping table. See “ 13.5.3.5 To Call Out to a Mapping Table from a Conversion Entry ” on page 443
X-REWRITE-SMS-ADDRESS		Used for local SMS address validity checks. See “ C.2.4 Site-defined Address Validity Checks and Translations ” on page 929

10.3.1

File Format in the Mappings File

The `mappings` file consists of a series of separate tables. Each table begins with its name. Names always have an alphabetic character in the first column. The table name is followed by a required blank line, and then by the entries in the table. Entries consist of zero or more indented lines. Each entry line consists of two columns separated by one or more spaces or tabs. Any spaces within an entry must be quoted using the \$ character. A blank line must appear after each mapping table name and between each mapping table; no blank lines can appear between entries in a single table. Comments are introduced by an exclamation mark (!) in the first column.

The resulting format looks like:

`TABLE1_NAME`

```
pattern1-1    template1-1
pattern1-2    template1-2
pattern1-3    template1-3
```

```
pattern1-n    template1-n

TABLE2_NAME

pattern2-1    template2-1
pattern2-2    template2-2
pattern2-3    template2-3

.
.
.

pattern2-n    template2-n
```

```
TABLE3_NAME
```

An application using the mapping table TABLE2_NAME would map the string pattern2-2 into whatever is specified by template2-2. Each pattern or template can contain up to 256 and 1024 characters respectively. The maximum size of a line in the mapping file is 4096 characters. There is no limit to the number of entries that can appear in a mapping (although excessive numbers of entries may consume huge amounts of CPU and can consume excessive amounts of memory). Long lines (over 252 characters) may be continued by ending them with a backslash (\). The white space between the two columns and before the first column may not be omitted.

Duplicate mapping table names are not allowed in the mappings file.

10.3.1.1

Including Other Files in the Mappings File

Other files may be included in the mappings file. This is done with a line of the form:

```
<file-spec
```

This effectively substitutes the contents of the file file-spec into the mappings file at the point where the include appears. The file specification should specify a full file path (directory, and so forth). All files included in this fashion must be world readable. Comments are also allowed in such included mappings file. Includes can be nested up to three levels deep. Include files are

loaded at the same time the `mappings` file is loaded—they are not loaded on demand, so there is no performance or memory savings involved in using include files.

10.3.2 Mapping Operations

All mappings in the `mappings` file are applied in a consistent way. The only things that change from one mapping to the next is the source of input strings and what the output from the mapping is used for.

A mapping operation always starts off with an input string and a mapping table. The entries in the mapping table are scanned one at a time from top to bottom in the order in which they appear in the table. The left side of each entry is used as pattern, and the input string is compared in a case-blind fashion with that pattern. For details about which MTA processes uses which tables and when, see [Table 10–2](#). This section consists of the following subsections:

- “[10.3.2.1 Mapping Entry Patterns](#)” on page 241
- “[10.3.2.2 IP Matching](#)” on page 243
- “[10.3.2.3 Mapping Entry Templates](#)” on page 244

10.3.2.1 Mapping Entry Patterns

Patterns can contain wildcard characters. In particular, the usual wildcard characters are allowed: an asterisk (*) matches zero or more characters, and each percent sign (%) matches a single character. Asterisks, percent signs, spaces, and tabs can be quoted by preceding them with a dollar sign (\$). Quoting an asterisk or percent sign robs it of any special meaning. Spaces and tabs must be quoted to prevent them from ending prematurely a pattern or template. Literal dollar sign characters should be doubled (\$\$), the first dollar sign quoting the second one.

TABLE 10–3 Mapping Pattern Wildcards

Wildcard	Description
%	Match exactly one character.
*	Match zero or more characters, with maximal or “greedy” left-to-right matching
Back match	Description
\$ n*	Match the nth wildcard or glob.
Modifiers	Description
\$ _	Use minimal or “lazy” left-to-right matching.
\$ @	Turn off “saving” of the succeeding wildcard or glob.

TABLE 10-3 Mapping Pattern Wildcards *(Continued)*

Glob wildcard	Description
\$^	Turn on “saving” of the succeeding wildcard or glob; this is the default.
\$A%	Match one alphabetic character, A-Z or a-z.
\$A*	Match zero or more alphabetic characters, A-Z or a-z.
\$B%	Match one binary digit (0 or 1).
\$B*	Match zero or more binary digits (0 or 1).
\$D%	Match one decimal digit 0-9.
\$D*	Match zero or more decimal digits 0-9.
\$H%	Match one hexadecimal digit 0-9 or A-F.
\$H*	Match zero or more hexadecimal digits 0-9 or A-F.
\$O%	Match one octal digit 0-7.
\$O*	Match zero or more octal digits 0--7.
\$S%	Match one symbol set character, for example, 0-9, A-Z, a-z, _, \$.
\$S*	Match zero or more symbol set characters, that is, 0-9, A-Z, a-z, _, \$.
\$T%	Match one tab or vertical tab or space character.
\$T*	Match zero or more tab or vertical tab or space characters.
\$X%	A synonym for \$H%.
\$X*	A synonym for \$H*.
\$[c]%	Match character c.
\$[c]*	Match arbitrary occurrences of character c.
\$[c ₁ c ₂ ... c _n]%	Match exactly one occurrence of character c ₁ , c ₂ , or c _n .
\$[c ₁ c ₂ ... c _n]*	Match arbitrary occurrences of any characters c ₁ , c ₂ , or c _n .
\$[c ₁ -c _n]%	Match any one character in the range c ₁ to c _n .
\$[c ₁ -c _n]*	Match arbitrary occurrences of characters in the range c ₁ to c _n .
\$<IPv4>	Match an IPv4 address, ignoring bits.
\$IPv4	Match an IPv4 address, keeping prefix bits.
\$IPv6	Match an IPv6 address. Note that IPv6 connection handling is not currently supported in Messaging Server.

Within globs, that is, within a \$[...] construct, the backslash character, (\) is the quote character. To represent a literal hyphen, -, or right bracket,], within a glob the hyphen or right bracket must be quoted with a backslash.

All other characters in a pattern just represent and match themselves. In particular, single and double quote characters as well as parentheses have no special meaning in either mapping patterns or templates; they are just ordinary characters. This makes it easy to write entries that correspond to illegal addresses or partial addresses.

To specify multiple modifiers, or to specify modifiers and a back match, the syntax uses just one dollar character. For instance, to back match the initial wild card, without saving the back match itself, one would use \${@}0, not \${@}\$.0.

Note that the `imsimta test -match` utility may be used to test mapping patterns and specifically to test wildcard behavior in patterns.

Asterisk wildcards maximize what they match by working from left to right across the input string. For instance, when the input string `a/b/c` is compared to the pattern `*/*`, the left asterisk matches `a/b` and the right asterisk matches the remainder, `c`.

The `$_` modifier causes wildcard matching to be minimized, where the least possible match is considered the match, working from left to right across the pattern. For instance, when the string `a/b/c` is compared to the pattern `$_*/$_*`, the left `$_*` matches `a` and the right `$_*` matches `b/c`.

10.3.2.2 IP Matching

With IPv4 prefix matching, an IP address or subnet is specified, optionally followed by a slash and the number of bits from the prefix that are significant when comparing for a match. For example, the following matches anything in the `123.45.67.0` subnet:

```
$(123.45.67.0/24)
```

With IPv4 ignore bits matching, an IP address or subnet is specified, optionally followed by a slash and the number of bits to ignore when checking for a match. For example, the following matches anything in the `123.45.67.0` subnet:

```
$<123.45.67.0/8>
```

The following example matches anything in the range `123.45.67.4` through `123.45.67.7`:

```
$<123.45.67.4/2>
```

IPv6 matching matches an IPv6 address or subnet. Note that IPv6 connection handling is not currently supported in Messaging Server.

10.3.2.3

Mapping Entry Templates

If the comparison of the pattern in a given entry fails, no action is taken; the scan proceeds to the next entry. If the comparison succeeds, the right side of the entry is used as a template to produce an output string. The template effectively causes the replacement of the input string with the output string that is constructed from the instructions given by the template.

Almost all characters in the template simply produce themselves in the output. The one exception is a dollar sign (\$).

A dollar sign followed by a dollar sign, space, or tab produces a dollar sign, space, or tab in the output string. Note that all these characters must be quoted in order to be inserted into the output string.

A dollar sign followed by a digit *n* calls for a substitution; a dollar sign followed by an alphabetic character is referred to as a “metacharacter.” Metacharacters themselves do not appear in the output string produced by a template, but produce some special substitution or processing. See [Table 10–4](#) for a list of the special substitution and standard processing metacharacters. Any other metacharacters are listed in the sections about specific mappings tables,. See [Table 18–1](#).

TABLE 10–4 Mapping Template Substitutions and Metacharacters

Substitution sequence	Substitutes
\$n	The <i>n</i> th wildcarded field as counted from left to right starting from 0.
\$#...#	Sequence number substitution.
\$]...[URL lookup; substitute in result.
\$...	Applies specified mapping table to supplied string.
\${...}	General database substitution.
\$} <i>domain,attribute</i> {	Adds the capability to access per-domain attributes. <i>domain</i> is the domain in question and <i>attribute</i> is the attribute associated with the domain. If the domain exists and has the attribute, its initial value is substituted into the mapping result; if either the attribute or the domain does not exist, the mapping entry fails. <i>attributes</i> can be domain LDAP attributes or the special attributes defined below: __base_dn__ - The base DN for user entries in the domain __domain_dn__ - The DN of the domain entry itself __domain_name__ - The name of the domain (as opposed to an alias) __canonical_name__ - The canonical name associated with the domain
\$[...]	Invokes site-supplied routine; substitute in result.
Metacharacter	Description

TABLE 10-4 Mapping Template Substitutions and Metacharacters *(Continued)*

Substitution sequence	Substitutes
\$C	Continues the mapping process starting with the next table entry; uses the output string of this entry as the new input string for the mapping process. See “ Processing Control (\$C, \$L, \$R, \$E) ” on page 246
\$E	Ends the mapping process now; uses the output string from this entry as the final result of the mapping process. \$+1E exits immediately without interpreting the rest of the template. See “ Processing Control (\$C, \$L, \$R, \$E) ” on page 246
\$L	Continues the mapping process starting with the next table entry; use the output string of this entry as the new input string; after all entries in the table are exhausted, makes one more pass, starting with the first table entry. A subsequent match may override this condition with a \$C, \$E, or \$R metacharacter. See “ Processing Control (\$C, \$L, \$R, \$E) ” on page 246
\$R	Continues the mapping process starting with the first entry of the mapping table; uses the output string of this entry as the new input string for the mapping process. See “ Processing Control (\$C, \$L, \$R, \$E) ” on page 246
\$nA	Inserts the nth left character of the current address starting from position 0. The entire address is inserted if n is omitted.
\$nX	Inserts the nth left component of the mailhost starting from 0. The entire mailhost is inserted if n is omitted.
\$?x?	Mapping entry succeeds x percent of the time.
\$\	Forces subsequent text to lowercase.
\$^	Forces subsequent text to uppercase.
\$_	Leaves subsequent text in its original case.
\$=	Forces subsequent substituted characters to undergo quoting appropriate for insertion into LDAP search filters.
\$:x	Match only if the specified flag is set.
\$;x	Match only if the specified flag is clear.

This section consists of the following subsections:

- “[Wildcard Field Substitutions \(\\$n\)](#)” on page 246
- “[Controlling Text Case \(\\$\,,\\$^,\\$_\)](#)” on page 246
- “[Processing Control \(\\$C, \\$L, \\$R, \\$E\)](#)” on page 246
- “[Check for Special Flags](#)” on page 247
- “[Entry Randomly Succeeds or Fails \(\\$?x?\)](#)” on page 247
- “[Sequence Number Substitutions \(\\$#...#\)](#)” on page 248
- “[URL substitutions, \\$\]...\[](#)” on page 249
- “[Mapping Table Substitutions \(\\$|...|\)](#)” on page 249

- “General Lookup Table or Database Substitutions (\${...})” on page 250
- “Site-Supplied Routine Substitutions (\$[...])” on page 250
- “Generate UTF-8 Strings” on page 251

Wildcard Field Substitutions (\$n)

A dollar sign followed by a digit n is replaced with the material that matched the *n*th wildcard in the pattern. The wildcards are numbered starting with 0. For example, the following entry would match the input string PSI%A: :B and produce the resultant output string b@a.psi.siroe.com:

```
PSI$%*: :* $1@$0.psi.siroe.com
```

The input string PSI%1234: :USER would also match producing USER@1234.psi.siroe.com as the output string. The input string PSIABC: :DEF would not match the pattern in this entry and no action would be taken; that is, no output string would result from this entry.

Controlling Text Case (\$\, \$^, \$_)

The metacharacter \$\backslash forces subsequent text to lowercase, \$^ forces subsequent text to uppercase, and \$_ causes subsequent text to retain its original case. For instance, these metacharacters may be useful when using mappings to transform addresses for which case is significant.

Processing Control (\$C, \$L, \$R, \$E)

The \$C, \$L, \$R, and \$E metacharacters control whether and when the mapping process terminates. As described in “[10.3.2 Mapping Operations](#)” on page 241, the mapping process normally consists of a single pass, from top to bottom, ending at the first entry with a pattern matching the input string, with the result specified by the template of that entry. Even if the processing of a metacharacter in the template fails, the mapping process would normally terminate because the input string matched the pattern of that entry. The metacharacters \$C, \$L, and \$R cause the mapping process to continue instead of terminate. The metacharacter \$E explicitly terminates the mapping process, thus overriding a previous \$C on the same line.

The metacharacters \$C, \$L, and \$R can be used to change the input string and continue the mapping process. If the template modifies the output string and \$C, \$L, or \$R are used, the mapping process continues with the output string of the current entry used as the input string for further operations. \$C continues with the next entry. \$L continues with the next entry, but if no matching entry is found before the end of the table, one more pass will be made starting again at the top of the table. \$R continues from the top of the table.

Templates are processed from left to right. When the intention is that an entry should succeed and terminate the mapping process if the template succeeds, but that the mapping process should continue if the template fails, then the entry should use the \$C (or \$R or \$L) metacharacter to the left of the part that may fail and \$E to the right of that part.

For example, see the PORT_ACCESS table where the \$|...| callout is used in “[18.7.4 To Use DNS Lookups Including RBL Checking for SMTP Relay Blocking](#)” on page 563 to check the client IP address against the INTERNAL_IP table. If the lookup succeeds, the template processing ends with \$Y and the mapping process is terminated by the \$E. But if the lookup fails, the template processing ends with the failure, but the \$C causes the mapping process to continue. Because this template generated no output string, the original input string remains unmodified.

Because metacharacters \$L and \$R reiterate the mapping process, the number of iterative passes through a mapping table is limited to prevent infinite loops. A counter is incremented each time a pass is restarted with a new input string that is the same length or longer than the previous pass. If the resulting output string has a shorter length than the input string, the counter is reset to zero. A request to reiterate a mapping is ignored after the counter has exceeded 10.

Note that any of the metacharacters \$C, \$E, \$L, or \$R, when present in the template, influences the mapping process and control whether it terminates or continues. That is, it is possible to set up iterative mapping table entries, where the output of one entry becomes the input of another entry. If the template of a matching pattern does not contain any of the metacharacters \$C, \$E, \$L, or \$R, then \$E (immediate termination of the mapping process) is assumed.

The number of iterative passes through a mapping table is limited to prevent infinite loops. A counter is incremented each time a pass is restarted with a pattern that is the same length or longer than the previous pass. If the string has a shorter length than previously, the counter is reset to zero. A request to reiterate a mapping is not honored after the counter has exceeded 10.

Check for Special Flags

Some mapping probes have special flags set. These are flags that can be set, and then their presence/absence tested—for using the general mapping table facility of \$:, \$; tests. \$:x causes an entry to match only if the flag x is set. \$;x causes an entry to match only if the flag x is clear. See specific mapping table descriptions [Table 10–2](#) for any special flags that may apply for that table.

When the intention is that an entry should succeed and terminate if the flag check succeeds, but that the mapping process should continue if the flag check fails, then the entry should use the \$C metacharacter to the left of the flag check and use the \$E flag to the right of the flag check.

Entry Randomly Succeeds or Fails (\$?x?)

The metacharacters \$?x? in a mapping table entry cause the entry to “succeed” x percent of the time; the rest of the time, the entry “fails” and the output of the mapping entry’s input is taken unchanged as the output. (Note that, depending upon the mapping, the effect of the entry failing is not necessarily the same as the entry not matching in the first place.) The x should be a real number specifying the success percentage.

For instance, suppose that a system with IP address 123.45.6.78 is sending your site just a little too much SMTP email and you’d like to slow it down; you can use a PORT_ACCESS mapping table in the following way. Suppose you’d like to allow through only 25 percent of its connection

attempts and reject the other 75 percent of its connection attempts. The following PORT_ACCESS mapping table uses \$?25? to cause the entry with the \$Y (accept the connection) to succeed only 25 percent of the time; the other 75 percent of the time, when this entry fails, the initial \$C on that entry causes the MTA to continue the mapping from the next entry, which causes the connection attempt to be rejected with an SMTP error and the message: Try again later.

PORT_ACCESS

```
TCP | * | 25 | 123.45.6.78 | *           $C$?25?$Y  
TCP | * | 25 | 123.45.6.78 | *           $N45s$ 4.40$ Try$ again$ later
```

Sequence Number Substitutions (\$#...#)

A \$#...# substitution increments the value stored in an MTA sequence file and substitutes that value into the template. This can be used to generate unique, increasing strings in cases where it is desirable to have a unique qualifier in the mapping table output; for instance, when using a mapping table to generate file names.

Permitted syntax is any one of the following:

```
$#seq-file-spec| radix| width| m#  
##seq-file-spec| radix| width#  
##seq-file-spec| radix#  
##seq-file-spec#
```

The required *seq-file-spec* argument is a full file specification for an already existing MTA sequence file. The optional *radix* and *width* arguments specify the radix (base) in which to output the sequence value, and the number of digits to output, respectively. The default radix is 10. Radices in the range -36 to 36 are also allowed; for instance, base 36 gives values expressed with digits 0,...,9,A,...,Z. By default, the sequence value is printed in its natural width, but if the specified width calls for a greater number of digits, then the output is padded with 0s on the left to obtain the correct number of digits. Note that if a width is explicitly specified, then the radix must be explicitly specified also.

The optional *m* argument is a modulus. If this fourth argument is specified the value inserted is the sequence number retrieved from the file mod *m*. The default is not to perform any modulus operation.

As noted above, the MTA sequence file referred to in a mapping must already exist. To create an MTA sequence file, use the following UNIX command:

```
touch seq-file-spec
```

or

```
cat >seq-file-spec
```

A sequence number file accessed using a mapping table must be world readable in order to operate properly. You must also have an MTA user account (configured to be nobody in the `imta_tailor` file) in order to use such sequence number files.

URL substitutions, \$]...[

A substitution of the form `$]url[` is specially handled. `url` can be any supported URL type including `file:` and `data:`. Standard LDAP URLs can also be used, with the host and port omitted; the host and part are instead specified with the `LDAP_HOST` and `LDAP_PORT` options. That is, the LDAP URL should be specified as:

```
ldap:///dn[?attributes[?scope?filter]]
```

where the square bracket characters [and] shown above indicate optional portions of the URL. The `dn` is required and is a distinguished name specifying the search base. The optional `attributes`, `scope`, and `filter` portions of the URL further refine the information to return. That is, `attributes` specifies the attribute or attributes to be returned from LDAP directory entries matching this LDAP query. The `scope` may be any of `base` (the default), `one`, or `sub`. `filter` describes the characteristics of matching entries.

Certain LDAP URL substitution sequences are available for use within the LDAP query URL. The length of URLs can be 1024 characters. This also applies to expressions created by mappings and mapping calls to other mappings.

Mapping Table Substitutions (\$|...|)

A substitution of the form `$|mapping;argument|` is handled specially. The MTA looks for an auxiliary mapping table named `mapping` in the MTA `mappings` file, and uses `argument` as the input to that named auxiliary mapping table. The named auxiliary mapping table must exist and must set the `$Y` flag in its output if it is successful; if the named auxiliary mapping table does not exist or doesn't set the `$Y` flag, then that auxiliary mapping table substitution fails and the original mapping entry is considered to fail: the original input string is used as the output string.

Note that when you want to use processing control metacharacters such as `$C`, `$R`, or `$L` in a mapping table entry that does a mapping table substitution, the processing control metacharacter should be placed to the left of the mapping table substitution in the mapping table template; otherwise the “failure” of a mapping table substitution means that the processing control metacharacter is not seen.

General Lookup Table or Database Substitutions (\${...})

A substitution of the form \${text} is handled specially. The *text* part is used as a key to access the general lookup table or database (see “[10.9.1 MTA Text Databases](#)” on page 265 for more information). If *text* is found in the table, the corresponding template from the table is substituted. If *text* does not match an entry in the table, the input string is used unchanged as the output string.

If you are using the general lookup table you need to set the low order bit of the MTA option `use_text_databases`. That is, set it to an odd number. Changes to the `general.txt` need to be compiled into the MTA configuration using the `imsimta cnbuild` to compile and `imsimta reload` to reload the reloadable data.

If you are using a general database, it should be world readable to insure that it operates properly.

When you want to use processing control metacharacters such as \$C, \$R, or \$L in a mapping table entry that does a general table substitution, the processing control metacharacter should be placed to the left of the general table substitution in the mapping table template; otherwise the “failure” of a general table substitution means that the processing control metacharacter is not seen.

Site-Supplied Routine Substitutions (\$[...])

A substitution of the form \${[image, routine, argument]} is handled specially. The *image*, *routine*, *argument* part is used to find and call a customer-supplied routine. At runtime on UNIX, the MTA uses `dlopen` and `dlsym` to dynamically load and call the routine *routine* from the shared library *image*. The routine *routine* is then called as a function with the following argument list:

```
status = routine (argument, arglength, result, reslength)
```

The *argument* and *result* are 252-byte long character string buffers. The *argument* and *result* are passed as a pointer to a character string (for example, in C, as `char*`). The *arglength* and *reslength* are signed, long integers passed by reference. On input, *argument* contains the *argument* string from the mapping table template, and *arglength* the length of that string. On return, the resultant string should be placed in *result* and its length in *reslength*. This resultant string then replaces the \${[image, routine, argument]} in the mapping table template. The *routine* routine should return 0 if the mapping table substitution should fail and -1 if the mapping table substitution should succeed. If the substitution fails, then normally the original input string is used unchanged as the output string.

If you want to use processing control metacharacters such as \$C, \$R, or \$L in a mapping table entry that does a site-supplied routine substitution, you place the processing control metacharacter to the left of the site-supplied routine substitution in the mapping table template; otherwise, the “failure” of a mapping table substitution means that the processing control metacharacter is not seen.

The site-supplied routine callout mechanism allows the MTA's mapping process to be extended in all sorts of complex ways. For example, in a PORT_ACCESS or ORIG_SEND_ACCESS mapping table, a call to some type of load monitoring service could be performed and the result used to decide whether or not to accept a connection or message.

The site-supplied shared library image `image` should be world readable.

Generate UTF-8 Strings

You can generate UTF-8 strings from Unicode character values in the general mapping table facility. A Unicode metacharacter sequence of the form:

```
$&A0A0,20,A1A1&
```

produces a UTF-8 string containing the characters at position A0A0, 20, and A1A1 in it.

10.4 Other MTA Configuration Files

In addition to the `imta.cnf` file, Messaging Server provides several other configuration files to help you configure MTA services. These files are summarized in [Table 10–5](#). This section consists of the following subsections:

- “[10.4.1 Alias File](#)” on page 252
- “[10.4.2 TCP/IP \(SMTP\) Channel Option Files](#)” on page 253
- “[10.4.3 Conversion File](#)” on page 253
- “[10.4.4 Dispatcher Configuration File](#)” on page 253
- “[10.4.5 Mappings File](#)” on page 254
- “[10.4.6 Option File](#)” on page 255
- “[10.4.7 Tailor File](#)” on page 255
- “[10.4.8 Job Controller File](#)” on page 256

If you make changes to the reverse, forward, or general databases, then issue the command `imsimta reload` to get the changes to take effect (see “[10.9.1 MTA Text Databases](#)” on [page 265](#)). If you make changes to the `imta.cnf`, `mappings` file, `aliases`, `conversions`, or `option.dat` files that do not affect the `job_controller`, then you should issue an `imsimta cnbuild` followed by an `imsimta restart smtp`. If you make changes to `dispatcher.cnf` you need to do an `imsimta restart dispatcher`. If you make changes to the configuration files that are included in the compiled configuration that affect the job controller, but not the SMTP server, in many cases you should issue the following commands: `imsimta cnbuild` and `imsimta restart job_controller`.

See “[MTA Commands](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for details on these commands.

TABLE 10-5 MTA Configuration Files

File	Description
“10.4.1 Alias File” on page 252 (mandatory)	Implements aliases not present in the directory. <i>msg-svr-base/config/aliases</i>
“10.4.2 TCP/IP (SMTP) Channel Option Files” on page 253 SMTP Option Files)	Sets channel-specific options. <i>msg-svr-base/config/channel_option</i>
“10.4.3 Conversion File” on page 253	Used by the conversion channel to control message body part conversions. <i>msg-svr-base/config/conversions</i>
“10.4.4 Dispatcher Configuration File” on page 253 (mandatory)	Configuration file for the Dispatcher. <i>msg-svr-base/config/dispatcher.cnf</i>
“10.4.8 Job Controller File” on page 256 (mandatory)	Configuration file used by the Job Controller. <i>/msg-svr-base/config/job_controller.cnf</i>
MTA Configuration File (mandatory)	Used for address rewriting and routing as well as channel definition. <i>/msg-svr-base/config/imta.cnf</i>
“10.3 Mappings File” on page 237 (mandatory)	Repository of mapping tables. <i>/msg-svr-base/config/mappings</i>
“10.4.6 Option File” on page 255	File of global MTA options. <i>/msg-svr-base/config/option.dat</i>
“10.4.7 Tailor File” on page 255 (mandatory)	File to specify locations and some tuning parameters. <i>/msg-svr-base/config/imta_tailor</i>
General Lookup Table (optional)	General lookup facility is equivalent to the general database. Part of reloadable compiled configuration. File to specify locations and some tuning parameters. <i>/msg-svr-base/config/general.txt</i>
Forward Lookup Table (optional)	Lookup facility for To: addresses. Equivalent to forward database. Part of reloadable compiled configuration. <i>/msg-svr-base/config/forward.txt</i>
Reverse Lookup Table (optional)	Reverse lookup facility for From: addresses. Equivalent to reverse database. Part of reloadable compiled configuration. <i>/msg-svr-base/config/reverse.txt</i>

10.4.1

Alias File

The alias file, `aliases`, sets aliases not set in the directory. In particular, the address for root is a good example. Aliases set in this file will be ignored if the same aliases exist in the directory. For more information about aliases and the `aliases` file, see “[10.5 Aliases](#)” on page 261.

After making changes to the `aliases` file, you must restart the MTA for the changes to take effect..

10.4.2 TCP/IP (SMTP) Channel Option Files

TCP/IP channel option files control various characteristics of TCP/IP channels. Channel option files must be stored in the MTA configuration directory and named `x_option`, where `x` is the name of the channel. For example, `msg-svr-base/config/tcp_local_option`. For more information refer to “[12.4.1 Configuring SMTP Channel Options](#)” on page 360. For complete information on all channel option keywords and syntax, see the *Sun Java System Messaging Server 6.3 Administration Reference*.

10.4.3 Conversion File

The conversion file, `conversions`, specifies how the conversion channel performs conversions on messages flowing through the MTA. Any subset of MTA traffic can be selected for conversion and any set of programs or command procedures can be used to perform conversion processing. The MTA looks at the conversion file to choose an appropriate conversion for each body part.

For more information about the syntax of this file, see “[13.5 The Conversion Channel](#)” on page 431

10.4.4 Dispatcher Configuration File

The Dispatcher configuration file, `dispatcher.cnf`, specifies Dispatcher configuration information. A default configuration file is created at installation time and can be used without any changes made. However, if you want to modify the default configuration file for security or performance reasons, you can do so by editing the `dispatcher.cnf` file. (For conceptual information, see “[8.3 The Dispatcher](#)” on page 195)

The Dispatcher configuration file format is similar to the format of other MTA configuration files. Lines specifying options have the following form:

`option=value`

The `option` is the name of an option and `value` is the string or integer to which the option is set. If the `option` accepts an integer value, a base may be specified using notation of the form `b%v`, where `b` is the base expressed in base 10 and `v` is the actual value expressed in base `b`. Such option specifications are grouped into sections corresponding to the service to which the following option settings apply, using lines of the following form:

`[SERVICE=service-name]`

The service-name is the name of a service. Initial option specifications that appear before any such section tag apply globally to all sections.

The following is a sample Dispatcher configuration file (`dispatcher.cnf`).

```
! The first set of options, listed without a [SERVICE=xxx]
! header, are the default options that will be applied to all
! services.
!
MIN_PROCS=0
MAX_PROCS=5
MIN_CONNS=5
MAX_CONNS=20
MAX_LIFE_TIME=86400
MAX_LIFE_CONNS=100
MAX_SHUTDOWN=2
!
! Define the services available to Dispatcher
!
[SERVICE=SMTP]
PORT=25
IMAGE=msg-svr-base/lib/tcp_smtp_server
LOGFILE=msg-svr-base/log/tcp_smtp_server.log
```

For more information about the parameters for this file, see the [Sun Java System Messaging Server 6.3 Administration Reference](#).

10.4.5 Mappings File

The `mappings` file defines how the MTA maps input strings to output strings.

Many components of the MTA employ table lookup-oriented information. Generally speaking, this sort of table is used to transform (that is, map) an input string into an output string. Such tables, called mapping tables, are usually presented as two columns, the first (or left-hand) column giving the possible input strings and the second (or right-hand) column giving the resulting output string for the input it is associated with. Most of the MTA databases are instances of this type of mapping table. The MTA database files, however, do not provide wildcard-lookup facilities, owing to inherent inefficiencies in having to scan the entire database for wildcard matches.

The `mappings` file provides the MTA with facilities for supporting multiple mapping tables. Full wildcard facilities are provided, and multi-step and iterative mapping methods can be accommodated as well. This approach is more compute-intensive than using a database, especially when the number of entries is large. However, the attendant gain in flexibility may actually serve to eliminate the need for most of the entries in an equivalent database, and this may actually result in lower overhead overall.

You can test mapping tables with the `imsimta test -mapping` command. For more information about the syntax of the `mappings` file and the `test -mapping` command, see the “[10.3 Mappings File](#)” on page [237](#) and the [Sun Java System Messaging Server 6.3 Administration Reference](#).

After making changes to the `mappings` file, you must restart the MTA or issue the command `imsimta reload`.

10.4.6 Option File

The options file, `option.dat`, specifies global MTA options as opposed to channel-specific options.

You can use the options file to override the default values of various parameters that apply to the MTA as a whole. In particular, the option file is used to establish sizes of the various tables into which the configuration and alias files are read. You can also use the options file to limit the size of messages accepted by the MTA, specify the number of channels allowed in the MTA configuration, set the number of rewrite rules allowed, and so on.

In `option.dat`, lines starting with #, ! or ; are treated as comment lines, even if the preceding line has a trailing \ meaning that it is being continued. This means that long options that may contain these characters, in particular delivery options, care has to be taken.

For delivery options, where a natural layout would lead to continuation lines starting with a # or !, there is a safe and neat work around.

For more information about the syntax of the options file, see the [Sun Java System Messaging Server 6.3 Administration Reference](#).

10.4.7 Tailor File

The tailor file, `imta_tailor`, sets the location of various MTA components. For the MTA to function properly, the `imta_tailor` file must always reside in the `msg-svr-base/config` directory.

Although you can edit this file to reflect the changes in a particular installation, you must do so with caution. After making any changes to this file, you must restart the MTA. It is preferable to make the changes while the MTA is down.

Note – Do not edit this file unless absolutely necessary.

For complete information on this file, see the [Sun Java System Messaging Server 6.3 Administration Reference](#).

10.4.8 Job Controller File

The Job Controller creates and manages channel jobs for delivering messages. These channel jobs run inside processing pools within the Job Controller. A pool can be thought of a “place” where the channel jobs are run. The pool provides a computing area where a set of jobs can operate without vying for resources with jobs outside of the pool. (For information on Job Controller concepts and channel keyword configuration, refer to “[8.7 The Job Controller](#)” on page 202, “[12.5.4 Processing Pools for Channel Execution Jobs](#)” on page 384 and “[12.5.5 Service Job Limits](#)” on page 384.)

The Job Controller file, `job_controller.cnf`, specifies the following channel processing information:

- Defines various pools
- Specifies for all channels, the master program name and the slave program name, if applicable

In the `imta.cnf` file, you can specify the name of a process pool (that was defined in `job_controller.cnf`) by using the `pool` keyword. For example, the following fragment from a sample `job_controller.cnf` file defines the pool `MY_POOL`:

```
[POOL=MY_POOL]
job_limit = 12
```

The following fragment from a sample `imta.cnf` file specifies the pool `MY_POOL` in a channel block:

```
channel_x pool MY_POOL
channel_x-daemon
```

If you want to modify the parameters associated with the default pool configuration or add additional pools, you can do so by editing the `job_controller.cnf` file, then stopping and restarting the Job Controller.

The first pool in the Job Controller configuration file is used for any requests that do not specify the name of a pool. The MTA channels defined in the MTA configuration file (`imta.cnf`) can have their processing requests directed to a specific pool by using the `pool` channel keyword followed by the name of the pool. The pool name must match the name of a pool in the Job Controller configuration. If the Job Controller does not recognize the requested pool name, the request is ignored.

In the initial configuration, the following pools are defined: `DEFAULT`, `LOCAL_POOL`, `IMS_POOL`, `SMTP_POOL`.

10.4.8.1 Examples of Use

Typically, you would add additional pool definitions to the Job Controller configuration if you wanted to differentiate processing of some channels from that of other channels. You might also choose to use pools with different characteristics. For example, you might need to control the number of simultaneous requests that some channels are allowed to process. You can do this by creating a new pool with the job limit, then use the pool channel keyword to direct those channels to the new, more appropriate pool.

In addition to the definition of pools, the Job Controller configuration file also contains a table of the MTA channels and the commands that the Job Controller must use to process requests for each channel. The two types of requests are termed “master” and “slave.” Typically, a channel master program is invoked when there is a message stored in an MTA message queue for the channel. The master program dequeues the message.

A slave program is invoked to poll a channel and pick up any messages inbound on that channel. While nearly all MTA channels have a master program, many do not have or need a slave program. For example, a channel that handles SMTP over TCP/IP doesn’t use a slave program because a network service, the SMTP server, receives incoming SMTP messages upon request by any SMTP server. The SMTP channel’s master program is the MTA’s SMTP client.

If the destination system associated with the channel cannot handle more than one message at a time, you need to create a new type of pool whose job limit is one:

```
[POOL=single_job]
job_limit=1
```

On the other hand, if the destination system has enough parallelism, you can set the job limit to a higher value.

[Example 10–1](#) shows a sample Job Controller configuration file. [Table 10–6](#) shows the available options.

EXAMPLE 10–1 Sample Job Controller Configuration File in UNIX

```
!MTA Job Controller configuration file
!
!Global defaults
tcp_port=27442      (1)
secret=never mind
slave_command=NULL    (2)
max_life_age=3600     (3)
!
!
!Pool definitions
!
[POOL=DEFAULT]        (4)
```

EXAMPLE 10-1 Sample Job Controller Configuration File in UNIX *(Continued)*

```
job_limit=10          (5)
!
[POOL=LOCAL_POOL]
job_limit=10
!
[POOL=IMS_POOL]
job_limit=1
!
[POOL=SMTP_POOL]
job_limit=1
!
!Channel definitions
!
!
[CHANNEL=l]          (6)
master_command=msg-svr-base/lib/l_master
!
[CHANNEL=ims-ms]
master_command=msg-svr-base/lib/ims_master
!
[CHANNEL=tcp_*]       (7)
master_command=msg-svr-base/lib/tcp_smtp_client
```

The key items in the preceding example (numbered, enclosed in parentheses, and in bold font) are:

1. This global option defines the TCP port number on which the Job Controller listens for requests.
2. Sets a default **SLAVE_COMMAND** for subsequent [CHANNEL] sections.
3. Sets a default **MAX_LIFE_AGE** for subsequent [CHANNEL] sections.
4. This [POOL] section defines a pool named **DEFAULT**.
5. Set the **JOB_LIMIT** for this pool to 10.
6. This [CHANNEL] section applies to a channel named **l**, the UNIX local channel. The only definition required in this section is the **master_command**, which the Job Controller issues to run this channel. Since no wildcard appears in the channel name, the channel must match exactly.
7. This [CHANNEL] section applies to any channel whose name begins with **tcp_***. Since this channel name includes a wildcard, it will match any channel whose name begins with **tcp_**.

Example of Adding Additional Pools

The Job Controller creates and manages channel jobs for delivering messages. These channel jobs run inside processing pools within the Job Controller. A pool can be thought of a “place” where the channel jobs are run. The pool provides a computing area where a set of jobs can operate without vying for resources with jobs outside of the pool. Note that the job limit that is set in the `job_controller` is per pool. So, for example, if you have your `SMTP_POOL` defined with a `job_limit` of 10, then only 10 `tcp_smtp` client processes can run in that pool at any given time.

There are situations where one may want to create additional `tcp_*` channels (say, for example, a `tcp` channel for particularly slow mail sites). It is a good idea to have these channels run in different pools. The reason for this is if we created ten different `tcp_*` channels and they were all running in `SMTP_POOL`, it is possible to only have one `tcp_smtp` client running per `tcp_*` channel at any given time (depending on whether or not there is mail destined for all the `tcp_*` channels and given that `SMTP_POOL` defined with a `job_limit` of 10). Assuming there is heavy load on the system and that all queues have messages waiting to go out the various `tcp_*` channels, this would not be efficient. It is much more likely that one would want to define additional pools for the additional `tcp_*` channels so that there is no contention for slots.

For example, suppose we set up the following `tcp_*` channels:

```
tcp_yahoo smtp mx pool yahoo_pool keyword keyword keyword
tcp-yahoo-daemon

tcp_aol smtp mx keyword keyword pool aol_pool
tcp-aol-daemon

tcp_hotmail smtp mx pool hotmail_pool keyword keyword keyword
tcp-hotmail-daemon
...
tcp_sun smtp mx pool sun_pool keyword keyword keyword
tcp-sun-daemon
```

In order to add have ten `tcp_smtp_client` processes for each of the new channels we would add the following in the `job_controller.cnf` file:

```
[POOL=yahoo_pool]
job_limit=10

[POOL=aol_pool]
job_limit=10

[POOL=hotmail_pool]
job_limit=10

...
```

```
[POOL=sun_pool]
job_limit=10
```

For more information about pools, see “[12.5.4 Processing Pools for Channel Execution Jobs](#)” on [page 384](#).

TABLE 10–6 Job Controller Configuration File Options

Option	Description
General Options	Description
INTERFACE_ADDRESS= <i>adapter</i>	Specifies the IP address interface to which the Job Controller should bind. The value specified (<i>adapter</i>) can be one of ANY, ALL, LOCALHOST, or an IP address. By default the Job Controller binds to all addresses (equivalent to specifying ALL or ANY). Specifying INTERFACE_ADDRESS=LOCALHOST means that the Job Controller only accepts connections from within the local machine. This does not affect normal operation, since no inter-machine operation is supported by the Job Controller. However, this may be inappropriate in an HA environment where an HA agent may be checking if the Job Controller is responding. If the machine on which the Messaging Server is running is in an HA environment, has an “internal network” adapter and an “external network” adapter, and you are not confident of your firewall’s ability to block connections to high port numbers, you should consider specifying the IP address of the “internal network” adapter.
MAX_MESSAGES= <i>integer</i>	The Job Controller keeps information about messages in an in-memory structure. In the event that a large backlog builds, it may need to limit the size of this structure. If the number of messages in the backlog exceeds the parameter specified here, information about subsequent messages is not kept in memory. Mail messages are not lost because they are always written to disk, but they are not considered for delivery until the number of messages known by the Job Controller drops to half this number. At this point, the Job Controller scans the queue directory mimicking an <code>imsimta cache -sync</code> command. The minimum value is 10. See “ 8.7 The Job Controller ” on page 202 for more information. The default is 100000.
SECRET= <i>file_spec</i>	Shared secret used to protect requests sent to the Job Controller.
SYNCH_TIME= <i>time_spec</i>	The Job Controller occasionally scans the queue files on disk to check for missing files. By default, this takes place every four hours, starting four hours after the Job Controller is started. The format of the <i>time_spec</i> is <i>HH:MM/hh:mm</i> or <i>/hh:mm</i> . The variable <i>hh:mm</i> is the interval between the events in hours (<i>h</i>) and minutes (<i>m</i>). The variable <i>HH:MM</i> is the first time in a day the even should take place. For example specifying, 15:45/7:15 starts the event at 15:45 and every seven hours and fifteen minutes from then.
TCP_PORT= <i>integer</i>	Specifies the TCP port on which the Job Controller should listen for request packets. Do not change this unless the default conflicts with another TCP application on your system. If you do change this option, change the corresponding IMTA_JBC_SERVICE option in the MTA tailor file, <code>msg-svr-base/config/imta_tailor</code> , so that it matches. The TCP_PORT option applies globally and is ignored if it appears in a [CHANNEL] or [POOL] section.

TABLE 10-6 Job Controller Configuration File Options *(Continued)*

Option	Description
Pool Option	Description
Channel Option	Description
<code>JOB_LIMIT=integer</code>	Specifies the maximum number of processes that the pool can use simultaneously (in parallel). The <code>JOB_LIMIT</code> applies to each pool individually; the maximum total number of jobs is the sum of the <code>JOB_LIMIT</code> parameters for all pools. If set outside of a section, it is used as the default by any <code>[POOL]</code> section that doesn't specify <code>JOB_LIMIT</code> . This option is ignored inside of a <code>[CHANNEL]</code> section.
<code>MASTER_COMMAND=file_spec</code>	Specifies the full path to the command to be executed by the UNIX system process created by the Job Controller to run the channel and dequeue messages outbound on that channel. If set outside of a section, it is used as the default by any <code>[CHANNEL]</code> section that doesn't specify a <code>MASTER_COMMAND</code> . This option is ignored inside of a <code>[POOL]</code> section.
<code>MAX_LIFE_AGE=integer</code>	Specifies the maximum life time for a channel master job in seconds. If this parameter is not specified for a channel, then the global default value is used. If no default value is specified, 14400 (240 minutes) is used.
<code>MAX_LIFE_CONNS=integer</code>	In addition to the maximum life age parameter, the life expectancy of a channel master job is limited by the number of times it can ask the Job Controller if there are any messages. If this parameter is not specified for a channel, then the global default value is used. If no default value is specified, 300 is used.
<code>SLAVE_COMMAND=file_spec</code>	Specifies the full path to the command to be executed by the UNIX system process created by the Job Controller in order to run the channel and poll for any messages inbound on the channel. Most MTA channels do not have a <code>SLAVE_COMMAND</code> . If that is the case, the reserved value <code>NULL</code> should be specified. If set outside of a section, it is used as the default by any <code>[CHANNEL]</code> section that doesn't specify a <code>SLAVE_COMMAND</code> . This option is ignored inside of a <code>[POOL]</code> section.

10.5 Aliases

The MTA provides a facility to support mailbox names associated with the local system that do not necessarily correspond to actual users: *aliases*. Aliases are useful for constructing mailing lists, forwarding mail, and providing synonyms for user names. For a discussion on how alias resolution is handled see “[9.1.2.2 The \\$V Metacharacter](#)” on page 210

Old-style mailing lists defined in the `aliases` file or `aliases` database now accept a nonpositional `[capture]` parameter. If used, the `[capture]` parameter specifies a capture address with the same semantics as capture addresses specified by the `LDAP_CAPTURE` attribute applied to a user or group in LDAP.

A value “`/`” given as an `[envelope_from]` nonpositional alias parameter, as an error to positional alias parameter, or as a value of the `mgrpErrorsTo` LDAP attribute, is now interpreted

as a request to revert to using the original envelope `From:` address for the incoming message while retaining mailing list semantics. This can be useful for setting up mailing lists that report all forms of list errors to the original sender.

10.5.1

The Alias Database

Use of the Alias Database is discouraged. Use the `aliases` file instead, especially since it can be dynamically reloaded using the `imsimta reload` command.

The MTA uses the information in the directory and creates the alias database. The alias database is consulted once each time the regular alias files is consulted. However, the alias database is checked before the regular alias file is used. In effect, the database acts as a sort of address rewriter that is invoked prior to using the alias file.

Note – The format of the database itself is private. Do not try to edit the database directly. Make all required changes in the directory.

10.5.2

The Alias File

The `aliases` file is used to set aliases not set in the directory. In particular, the postmaster alias is a good example. Aliases set in this file will be ignored, if the same aliases exist in the directory. Changes can be activated by issuing the `imsimta reload` command (or restarting the MTA). Any line that begins with an exclamation point is considered to be a comment and is ignored. Blank lines are also ignored.

Note – Messaging Server provides other facilities for address manipulation, such as the address reversal database and specialized mapping tables. For best performance, however, rewrite rules should be used whenever possible to perform address manipulations. See [Chapter 11, “Configuring Rewrite Rules.”](#)

A physical line in this file is limited to 1024 characters. You can split a logical line into multiple physical lines using the backslash (\) continuation character.

The format of the file is as follows:

`user@domain: address` (*for users in hosted domains*)

`user@domain: address` (*for users in non-hosted domains. Example: default-domain*)

For example:

```
! A /var/mail/ user
inetmail@siroe.com: inetmail@native-daemon

! A message store user
ms_testuser@siroe.com: mstestuser@ims-ms-daemon
```

10.5.3 Including Other Files in the Alias File

Other files can be included in the primary `aliases` file. A line of the following form directs the MTA to read the `file-spec` file:

```
<file-spec
```

The file specification must be a complete file path specification and the file must have the same protections as the primary `aliases` file; for example, it must be world readable.

The contents of the included file are inserted into the `aliases` file at its point of reference. The same effect can be achieved by replacing the reference to the included file with the file's actual contents. The format of include files is identical to that of the primary `aliases` file itself. Indeed, include files may themselves include other files. Up to three levels of include file nesting are allowed.

10.6 Command Line Utilities

Messaging Server provides several command-line utilities that enable you to perform various maintenance, testing, and management tasks for the MTA. For example, you use the `imsimta cnbuild` command to compile the MTA configuration, alias, mapping, security, system wide filter, and option files. For complete information on the MTA command-line utilities, see the [Sun Java System Messaging Server 6.3 Administration Reference](#).

10.7 SMTP Security and Access Control

For information about SMTP security and access control, see [Chapter 18, “Mail Filtering and Access Control”](#)

10.8 Log Files

All MTA specific log files are kept in the log directory, (*msg-svr-base/log*). This directory contains log files that describe message traffic through the MTA and log files that describe information about specific master or slave programs.

For more information about MTA log files, see [Chapter 25, “Managing Logging.”](#)

10.9 To Convert Addresses from an Internal Form to a Public Form

Addresses can be converted from an internal form to a public, advertised form using the Address-Reversal text database (also called the *reverse text database*) and the REVERSE mapping table. For example, while `uid@mailhost.siroe.com` might be a valid address within the `siroe.com` domain, it might not be an appropriate address to expose to the outside world. You may prefer a public address like `firstname.lastname@siroe.com`.

Messaging Server provides other facilities for address manipulation, such as the `aliases` file and specialized mapping tables. For best performance, however, rewrite rules should be used whenever possible to perform address manipulations. See [Chapter 11, “Configuring Rewrite Rules.”](#)

This section consists of the following subsections:

- [“10.9.1 MTA Text Databases” on page 265](#)
- [“10.9.2 To Set Address Reversal Controls” on page 266](#)
- [“10.9.3 The Forward Lookup Table and FORWARD Address Mapping” on page 268](#)

In the reverse text database, the public address for each user is specified by the `mail` attribute of the user entry in the directory.

The reverse text database contains a mapping between a valid address and a public address. See [“10.9.1 MTA Text Databases” on page 265](#) for more information.

If an address is found in the database, the corresponding right side from the database is substituted for the address. If the address is not found, an attempt is made to locate a mapping table named REVERSE in the `mappings` file. No substitution is made, and rewriting terminates normally if the table does not exist or no entries from the table match.

If a REVERSE mapping table is found in the `mappings` file, and if the address matches a mapping entry, the resulting string replaces the address if the entry specifies a `$Y`. A `$N` discards the result of the mapping. If the mapping entry specifies `$D` in addition to `$Y`, the resulting string runs through the reversal database once more; and if a match occurs, the template from the database replaces the mapping result (and hence the address). The form of general REVERSE mapping table entries (that is, entries that apply to all channels) is shown below. Note that flags can be either in front of the new address or at the end.

REVERSE

OldAddress \$Y[Flags]NewAddress

The form of *channel-specific* entries (that is, mapping that only occurs only on messages passing through a specific channel) is shown below. Note that you must set `use_reverse_database` to 13 in the `option.dat` for channel-specific entries to work.

REVERSE

source-channel|destination-channel|OldAddress \$Y[Flags]NewAddress

REVERSE mapping table flags as shown in Table 10–7.

TABLE 10–7 REVERSE mapping table flags

Flags	Description
\$Y	Use output as new address.
\$N	Address remains unchanged.
\$D	Run output through the reversal database.
\$A	Add pattern as reverse database entry.
\$F	Add pattern as forward database entry.
Flag comparison	Description
:\$B	Match only header (body) addresses.
:\$E	Match only envelope addresses.
:\$F	Match only forward pointing addresses.
:\$R	Match only backwards pointing addresses.
:\$I	Match only message-ids.

10.9.1 MTA Text Databases

MTA use of sleepycat databases is being deprecated because of the instability it introduces in Messaging Server deployments. (Note that sleepycat will not be removed in the near future). As a result, MTA text databases for the reverse, forward and general databases should be used instead.

To set up text databases:

1. Prepare a text file containing the data.

This is in the same format that `imsimta crdb` uses: one entry per line with two fields separated by one or more spaces. The file names are specified by the `IMTA_GENERAL_DATA`, `IMTA_REVERSE_DATA`, and `IMTA_FORWARD_DATA` options in `imta_tailor`, which normally point, respectively, to `IMTA_TABLE:general.txt`, `IMTA_TABLE:reverse.txt`, and `IMTA_TABLE:forward.txt` in `msg-svr-base/config/`.

`general.txt` - general database
`reverse.txt` - reverse database
`forward.txt` - forward database

2. Set the appropriate bit or bits in the `USE_TEXT_DATABASES` option:

bit 0 (value 1) - use text file for general database
bit 1 (value 2) - use text file for reverse database
bit 2 (value 4) - use text file for forward database

3. Set whatever additional options are needed to enable the desired databases.

For example, `USE_REVERSE_DATABASE`, `USE_FORWARD_DATABASE`, or whatever

4. Run `imsimta cnbuild`
5. Run `imsimta reload`

The only case where `USE_TEXT_DATABASES` is not appropriate is for highly dynamic data. In those cases, you may be better served by writing your own MTA plug-ins rather than by relying on the built-in database support.

If the text database is not appropriate, and you wish to use the `crdb` (Sleepycat) database support, then it may be possible, by structuring your database usage style and updating process appropriately, to use either `imsimta crdb` or `imsimta db` to update the database without recompiling, reloading, or restarting. However, for this to work you either have to be in a situation where you can only add or update existing entries, in which case you can use `imsimta crdb`. Otherwise, you have to have your data structured as a series of add/delete/change operations. If your data isn't structured this way--and it usually isn't--you're back to replacing the entire database when updating, which in this case, makes text databases preferable.

10.9.2

To Set Address Reversal Controls

The `reverse` and `no reverse` channel keywords, and the MTA options `USE_REVERSE_DATABASE` and `REVERSE_ENVELOPE` are used to control the specifics of when and how address reversal is applied. By default, the address reversal operation applies to all addresses, not just to backward pointing addresses.

Address reversal can be enabled or disabled by setting the value of the `REVERSE_ENVELOPE` system option (Default: 1-on, 0-off).

`noreverse` on the destination channel specifies that address reversal is not applied to addresses in messages. `reverse` specifies that address reversal is applied. See “[12.6.9 Enabling Channel-Specific Use of the Reverse Database](#)” on page 395 for details.

`USE_REVERSE_DATABASE` controls whether the MTA uses the address reversal text database and `REVERSE` mapping as a source of substitution addresses. A value of 0 means address reversal is not used with any channel. A value of 5 (default) specifies that address reversal is applied to all addresses—not just to backward pointing addresses—after they have been rewritten by the MTA address rewriting process. A value of 13 specifies that address reversal is applied to addresses with the `reverse` channel keyword—not just to backward pointing addresses—after they have been rewritten by the MTA address rewriting process. Further granularity of address reversal operation can be specified by setting the bit values of the `USE_REVERSE_DATABASE` option. See “[Option File Format and Available Options](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for details.

The `REVERSE_ENVELOPE` option controls whether or not address reversal is applied to envelope From addresses as well as message header addresses.

See the detailed descriptions of these options and keywords in the Sun Java System Messaging Server Administration Reference for additional information on their effects.

10.9.2.1

General Reverse Mapping Example

An example of a general `REVERSE` Mapping is as follows: suppose that the internal addresses at `siroe.com` are of the form `user@mailhost.siroe.com`. However, the user name space is such that `user@host1.siroe.com` and `user@host2.siroe.com` specify the same person for all hosts at `siroe.com`. The following `REVERSE` mapping may be used in conjunction with the address-reversal text database:

`REVERSE`

`*@*.siroe.com $0@siroe.com$Y$D`

In this example, addresses of the form `name@anyhost.siroe.com` would be changed to `name@siroe.com`. The `$D` metacharacter causes the address-reversal database to be consulted. The address-reversal text database should contain entries of the form:

`user@mailhost.siroe.com first.last@siroe.com`

10.9.2.2

Channel-Specific Reverse Mapping Example

By default, the address reversal text database is used if the routability scope is set to the mail server domains. An example of a channel-specific `REVERSE` mapping table entry would be as follows:

REVERSE

```
tcp_*|tcp_local|binky@macho.siroe.com      $D$YRebecca.Woods@siroe.com
```

This entry tells the MTA that for any mail with source channel of `tcp_*` going out the destination channel of `tcp_local` to change addresses of the form `binky@macho.siroe.com` to `Rebecca.Woods@siroe.com`

Note – To enable channel-specific reverse mapping, you must set `USE_REVERSE_DATABASE` option in `option.dat` to 13. (Default=5.)

10.9.3 The Forward Lookup Table and FORWARD Address Mapping

Address reversals are not applied to envelope To: addresses. The reasons for this omission are fairly obvious—envelope To: addresses are continuously rewritten and modified as messages proceed through the mail system. The entire goal of routing is to convert envelope To: addresses to increasingly system and mailbox-specific formats. The canonicalization functions of address reversal are entirely inappropriate for envelope To: addresses.

In any case, plenty of machinery is available in the MTA to perform substitutions on envelope To: addresses. The alias file, alias database and general lookup table, provide precisely this functionality.

The MTA also provides the forward lookup table and FORWARD mapping, used for special sorts of forwarding purposes, such as pattern-based forwarding, source-specific forwarding, or auto-registration of addresses. Note that the forward lookup table and FORWARD mapping are intended for use primarily for certain special sorts of address forwarding; most sorts of address forwarding, however, are better performed using one of the MTA's other forwarding mechanisms.

The various substitution mechanisms for envelope To: addresses provide functionality equivalent to the reversal lookup table, but none yet discussed provide functionality equivalent to the reverse mapping. And circumstances do arise where mapping functionality for envelope To: addresses is useful and desirable.

10.9.3.1 The FORWARD Mapping Table

The FORWARD mapping table provides this functionality of pattern based forwarding, and also provides a mechanism for source specific forwarding. If a FORWARD mapping table exists in the mapping file, it is applied to each envelope To: address. No changes are made if this mapping does not exist or no entries in the mapping match.

If the address matches a mapping entry, the result of the mapping is tested. The resulting string will replace the envelope To: address if the entry specifies a \$Y; a \$N will discard the result of the mapping. See [Table 10–8](#) for a list of additional flags.

TABLE 10–8 FORWARD Output Mapping Table Flags Description

Flag	Description
\$D	Run output through the rewriting process again
\$G	Run output through the forward lookup table, if forward lookup table use has been enabled
\$H	Disable further forward lookup table or FORWARD mapping lookups
\$I	Hold the message as a .HELD file
\$N	Address remains unchanged
\$Y	Use output as new address

The FORWARD mapping, if present, is consulted before any forward lookup table lookup. If a FORWARD mapping matches and has the flag \$G, then the result of the FORWARD mapping will be checked against the forward lookup table, if forward lookup table use has been enabled via the appropriate setting of USE_FORWARD_DATABASE. (Note that if channel specific forward lookup table use has been specified, then the source address and source channel will be prefixed to the result of the FORWARD mapping before looking up in the forward lookup table.) If a matching FORWARD mapping entry specifies \$D, then the result of the FORWARD mapping (and optional forward table lookup) will be run through the MTA address rewriting process again. If a matching FORWARD mapping entry specifies \$H, then no further FORWARD mapping or database lookups will be performed during that subsequent address rewriting (that resulting from the use of \$D).

The following input flags are now available in the FORWARD mapping. In the past they were only available to the various *_ACCESS mappings.

TABLE 10–9 FORWARD Input Mapping Table Flags Description

Flag	Description
\$A	SASL used to authenticate connection.
\$D	NOTIFY=DELAYS active for this recipient.
\$E	Incoming connection used ESMTP/EHLO.
\$F	NOTIFY=FAILURES active for this recipient.
\$L	Incoming connection used LMTP/LHLO.
\$S	NOTIFY=SUCCESSES active for this recipient.

TABLE 10-9 FORWARD Input Mapping Table Flags Description (Continued)

Flag	Description
\$T	SSL/TLS used to secure connection.

The example below illustrates the use of a complex REVERSE and FORWARD mapping. Suppose that a system or pseudo domain named am.sigurd.innosoft.com associated with the `mr_local` channel produces RFC 822 addresses of the general form:

"lastname, firstname"@am.sigurd.example.com

or

"lastname,firstname"@am.sigurd.example.com

Although these addresses are perfectly legal they often confuse other mailers which do not fully comply with RFC 822 syntax rules—mailers which do not handle quoted addresses properly, for instance. Consequently, an address format which does not require quoting tends to operate with more mailers. One such format is

firstname.lastname@am.sigurd.example.com

Example of a complex FORWARD and REVERSE mapping:

REVERSE

```
*|mr_local|*,,$ *@"am.sigurd.example.com $Y"$1,$ $2"@am.sigurd.example.com
*|mr_local|*,,*@"am.sigurd.example.com    $Y"$1,$ $2"@am.sigurd.example.com
*|*|*,,$ *@"am.sigurd.example.com        $Y$3.$2@am.sigurd.example.com
*|*|*,,*@"am.sigurd.example.com        $Y$3.$2@am.sigurd.example.com
*|mr_local|*. *@am.sigurd.example.com   $Y"$2,$ $1"@am.sigurd.example.com
*|*|*. @"am.sigurd.example.com          $Y$2.$3@am.sigurd.example.com
```

FORWARD

```
*,,$ *@"am.sigurd.example.com           $Y"$0,$ $1"@am.sigurd.example.com
*,,*@"am.sigurd.example.com            $Y"$0,$ $1"@am.sigurd.example.com
*. @"am.sigurd.example.com             $Y"$1,$ $0"@am.sigurd.example.com
```

So the goals of the sample mapping tables in the above example are threefold. (1) Allow any of these three address formats above to be used. (2) Present only addresses in the original format to the `mr_local` channel, converting formats as necessary. (3) Present only addresses in the new unquoted format to all other channels, converting formats as necessary. (The REVERSE mapping shown assumes that bit 3 in the MTA option `USE_REVERSE_DATABASE` is set.

10.9.3.2

The Forward Lookup Table

In cases where address forwardings need to be auto-registered or source specific, the forward lookup table is available. Note that use of the Forward lookup table for simple forwarding of messages is generally not appropriate; the aliases file or alias lookup table is a more efficient way to perform such forwarding. By default, the forward lookup table is not used at all; its use must be explicitly enabled via the USE_FORWARD_DATABASE option. Forward table lookups are performed after address rewriting and after alias expansion is performed, and after any FORWARD mapping is checked. If a forward table lookup succeeds, the resulting substituted address is then run through the MTA address rewriting process all over again.

There are two mechanisms available for the forward lookup table, an in-memory hash table or conventional text database. Unless the size of the table is prohibitively large then hash table is recommended. (1,000 is not prohibitively large, 100,000 is). The hash table is enabled by setting bit 2 (value 4) in the use_text_databaseS option as well as setting use_forward_database. The hash table is read from *msg-svr-base/configure/forward.txt*, compiled into the reloadable part of the configuration, and can be forced to be reloaded into the active MTA processes by the `imsimta reload` command.

The format of the source text file by default is expected to be:

```
user1@domain1 changedmailbox1@changeddomain1
user2@domain2 changedmailbox@changeddomain2
```

But if source specific use of the forward text database has been enabled by setting bit 2 of the USE_FORWARD_DATABASE option, then the source text file format expected is:

```
source-channel|source-address|original-address changed-address
```

For instance, an entry such as

```
tcp_limited|bob@blue.com|helen@red.com "helen of troy"@siroe.com
```

will map the To: address helen@red.com to “helen of troy”@siroe.com if and only if the message is coming from bob@blue.com and the enqueueing channel is `tcp_limited`.

See “[10.9.1 MTA Text Databases](#)” on page [265](#) for more information on the Forward Text database.

10.10 Controlling Delivery Status Notification Messages

Delivery status notifications or *status notifications* are email status messages sent by the MTA to the sender and, optionally, the postmaster. Messaging Server allows you to customize notification messages by content and language. You can also create different messages for each type of delivery status (for example, FAILED, BOUNCED, TIMEDOUT, etc.). In addition, you can create status notifications for messages originating from specific channels.

By default, status notifications are stored in the *msg-svr-base/config/locale/C* directory (specified by the *IMTA_LANG* setting in the *msg-svr-base/config/imta_tailor* file). The filenames are as follows:

return_bounced.txt, *return_delivered.txt* *return_header.opt*, *return_timedout.txt*,
return_deferred.txt, *return_failed.txt*, *return_prefix.txt*, *return_delayed.txt*,
return_forwarded.txt, *return_suffix.txt*.

Message text for *.txt files should be limited to 78 characters per line. Note that you shouldn't change these files directly since they'll be overwritten when the current version of Messaging Server is upgraded. If you wish to modify these files and use them as your only set of notification message template files (*return_*.txt*), copy the files to a new directory and edit them there. Then, set the *IMTA_LANG* option in the *imta_tailor* file to point to the new directory containing those templates. If you wish to have multiple sets of notification files (for example, a set for each language) then you will need to set up a *NOTIFICATION_LANGUAGE* mapping table.

This section consists of the following subsections:

- “10.10.1 To Construct and Modify Status Notifications” on page 272
- “10.10.2 To Customize and Localize Delivery Status Notification Messages” on page 274
- “10.10.3 Internationalization of Generated Notices” on page 276
- “10.10.4 Additional Status Notification Message Features” on page 277

10.10.1

To Construct and Modify Status Notifications

A single notification message is constructed from a set of three files: *return_prefix.txt* + *return_ActionStatus.txt* + *return_suffix.txt*

To customize or localize notifications, you would create a complete set of *return_*.txt* files for each locale and/or customization and store it in a separate directory. For example, you could have French notification files stored in one directory, Spanish for another, and notifications for a special unsolicited bulk email channel stored in a third.

Note – Sample files for French, German, and Spanish are included in this release. These files can be modified to suit your specific needs.

For double-byte languages such as Japanese, be sure to construct your text in Japanese, then view the text as if it was ASCII to check for % characters. If there are accidental % characters, then replace them with %%.

The format and structure of status notification message sets is described below.

1. *return_prefix.txt* provides appropriate header text as well as introductory material for the body. The default for US-english locale is as follows:

Content-type: text/plain; charset=us-ascii
 Content-language: EN-US

This report relates to a message you sent with the following header fields: %H

Non-US-ASCII status notification messages should change the charset parameter and Content-Language header value appropriately (for example, for French localized files the values would be ISO-8859-1 and fr). %H is a header substitution sequence defined in [Table 10-10](#).

2. return_<ActionStatus>.txt contains status-specific text. ActionStatus refers to a message's MTA status type. For example, the default text for return_failed.txt is as follows:

Your message cannot be delivered to the following recipients:
 %R

The default text for return_bounced.txt is:

Your message is being returned. It was forced to return by the postmaster.

The recipient list for this message was:

%R

3. return_suffix.txt contains concluding text. By default this file is blank.

TABLE 10-10 Notification Message Substitution Sequences

Substitutions	Definition
%H	Expands into the message's headers.
%C	Expands into the number of units ¹ the message has been queued.
%L	Expands into the number of units ¹ the message has left in the queue before it is returned.
%F	Expands into the number of units ¹ a message is allowed to stay in the queue.
%S [%s]	Expands to the letter S or s if the previously expanded numeric value was not equal to one. Example: "%C day%s" can expand to "1 day" or "2 days" depending on how many days the message has been queued.
%U [%u]	Expands into the time units Hour [hour] or Day [day], in use. Example: "%C %U%s" can expand to "2 days" or "1 hour" depending on how many days or hours the message has been queued, and the value of the MTA option RETURN_UNITS. If you have set RETURN_UNITS=1 (hours) and your site is using localized status notification messages, you will need to edit return_delayed.txt and return_timedout.txt and replace the word "days" with the word hours for all languages other than English. French, replace jour(s) with heure(s). German, replace Tag(e) with Stunde(n). Spanish, replace dia(s) with hora(s)
%R	Expands into the list of the message's recipients.
%%	(Note that the text is scanned byte by byte for substitutions sequences regardless of character set. Check for unintended % signs if you are using a double byte character set.)

TABLE 10-10 Notification Message Substitution Sequences *(Continued)*

Substitutions	Definition
¹ Units is defined by the RETURN_UNITS option in the MTA Options file and can be hours or days (default).	

10.10.2 To Customize and Localize Delivery Status Notification Messages

Delivery Status Notification Messages can be localized such that messages will be returned to different users in different languages. For example, French notifications could be returned to users who have expressed a preference for French.

Localizing or customizing status notification messages consists of two steps:

1. Create a set of localized/customized `return_*.txt` message files and store each set in a separate directory. This is described in “[10.10.1 To Construct and Modify Status Notifications](#)” on page 272
2. Set up a `NOTIFICATION_LANGUAGE` mapping table.

The `NOTIFICATION_LANGUAGE` mapping table (located in `msg-svr-base/config/mappings`) specifies the set of localized or customized notification message files to use depending upon attributes (for example: language, country, domain, or address) of the *originating message* (the message causing the notification to be sent).

The original sender’s message is parsed to determine status notification type, source channel, preferred language, return address and first recipient. Depending on how the table is constructed, a set of notification files is selected depending on one or more of these attributes.

The format of the `NOTIFICATION_LANGUAGE` mapping table is as follows. The sample entry line has been wrapped for typographic reasons. The actual entry should appear on one physical line.

NOTIFICATION_LANGUAGE

```
dsn-type-list|source-channel|preferred-language|return-address \
|first-recipient $Idirectory-spec
```

- `dsn-type-list` is a comma-separated list of delivery status notification types. If multiple types are specified, they must be separated by commas and without spaces (a space will terminate the pattern of the mapping table entry). The types are as follows:
 - `failed` - Generic permanent failure messages (for example, no such user). Uses the `return_failed.txt` file.
 - `bounced` - Notification message used in conjunction with manual “bounces.” Done by the postmaster. Uses the `return_bounced.txt` file.
 - `timedout` - The MTA has been unable to deliver the message within the time allowed for delivery. The message is now being returned. Uses the `return_timedout.txt` file.

- delayed - The MTA has been unable to deliver the message, but will continue to try to deliver it. Uses the `return_delayed.txt` file.
- deferred - Non-delivery notification similar to “delayed” but without an indication of how much longer the MTA will continue to try to deliver. Uses the `return_deferred.txt` file.
- forwarded - A delivery receipt was requested for this message, however, this message has now been forwarded to a system that does not support such receipts. Uses the `return_forwarded.txt` file.
- source-channel is the channel generating the notification message, that is, the channel at which the message is currently queued. For example, `ims-ms` for the message store’s delivery queue, `tcp_local` for outbound SMTP queues, etc.
- preferred-language is the language expressed in the message being processed (the message for which the notification is being generated). The sources for this information are first the `accept_language` field. If that doesn’t exist the `Preferred-language:` header field and `X-Accept-Language:` header field are used. For a list of standard language code values, refer to the file `msg-srv-base/config/languages.txt`

This field, if not empty, will be whatever the originator of the message specified for the `Preferred-language:` or `X-Accept-language:` header line. As such, you may find nonsense characters in this field.

- return-address is the envelope `From: address` of the originating message. This is the envelope address to which the notification message will be sent and hence a possible indicator of what language to use.
- first-recipient is the envelope `To: address` (the first one, if the message is failing to more than one recipient) to which the original message was addressed. For example, in the notification “your message to `dan@siroe.com` could not be delivered”—in this case `dan@siroe.com` is the envelope `To: address` being reported on.
- directory-spec is the directory containing the `return_*.txt` files to use if the mapping table probe matches. Note that a `$I` must precede the directory specification.

For instance, a site that stores French notification files (`return_*.txt`) in a directory `/lc_messages/table/notify_french/` and Spanish notification files in `return_*.txt` files in a directory `/lc_messages/table/notify_spanish/` might use a table as shown below. Note that each entry must start with one or more spaces, and that there can be no blank lines between entries.

NOTIFICATION_LANGUAGE

```
! Preferred-language: header value specified
!
  *|*|fr|*|*      $I/lc_messages/table/notify_french/
  *|*|es|*|*      $IIMTA_TABLE/notify_spanish/
  *|*|en|*|*      $I/imta/lang/
!
```

```
! If no Preferred-language value, then select notification based on the
! country code in the domain name. EX: PF=French Polynesia; BO=Bolivia
!
*|*|*|*.fr|*      $I/imta/table/notify_french/
*|*|*|*.fx|*      $I/imta/table/notify_french/
*|*|*|*.pf|*      $I/imta/table/notify_french/
*|*|*|*.tf|*      $I/imta/table/notify_french/
*|*|*|*.ar|*      $I/imta/table/notify_spanish/
*|*|*|*.bo|*      $I/imta/table/notify_spanish/
*|*|*|*.cl|*      $I/imta/table/notify_spanish/
*|*|*|*.co|*      $I/imta/table/notify_spanish/
*|*|*|*.cr|*      $I/imta/table/notify_spanish/
*|*|*|*.cu|*      $I/imta/table/notify_spanish/
*|*|*|*.ec|*      $I/imta/table/notify_spanish/
*|*|*|*.es|*      $I/imta/table/notify_spanish/
*|*|*|*.gp|*      $I/imta/table/notify_spanish/
*|*|*|*.gt|*      $I/imta/table/notify_spanish/
*|*|*|*.gy|*      $I/imta/table/notify_spanish/
*|*|*|*.mx|*      $I/imta/table/notify_spanish/
*|*|*|*.ni|*      $I/imta/table/notify_spanish/
*|*|*|*.pa|*      $I/imta/table/notify_spanish/
*|*|*|*.ve|*      $I/imta/table/notify_spanish/
```

Note – A default `mappings.locale` file is provided with the installation and will be included in the `mappings` file to enable notification language mapping. To disable notification language mapping, comment out the include line as follows:

```
! <IMTA_TABLE:mappings.locale
```

(Read the comments in the file and modify to suit your needs.)

10.10.3 Internationalization of Generated Notices

Two option files can be used for both the delivery status and message disposition notification. These are intended to make internationalization of generated notices more flexible. They are as follows:

`IMTA_LANG:return_option.dat` (DSN) `IMTA_LANG:disposition_option.dat` (MDN)

The options available for these files are described in [Table 10–11](#).

TABLE 10-11 Delivery Status and Message Disposition Notification Options

Option	Description
DAY (DSN)	The text to insert for a %U or %u substitution when RETURN_UNITS=0 (the default) is set. Note that no distinction is made between %U and %u (unlike the default case where English “Day” or “day”, respectively, would be substituted).
DIAGNOSTIC_CODE (DSN)	Override for the “Diagnostic code:” text used in the construction of the per-recipient section of the first part of a DSN. This field should be specified in the same charset that’s used for the first part of the DSN.
HOUR (DSN)	The text to insert for a %U or %u substitution when RETURN_UNITS=1 is set. Note that no distinction is made between %U and %u (unlike the default case where English “Hour” or “hour”, respectively, would be substituted).
n.n.n (DSN)	When constructing the per-recipient part of a DSN a check will be made to see if there’s an option whose name matches the numeric per-recipient status. If there is the corresponding text will be inserted into the DSN. Additionally, if the REASON option specified above produces a zero length result the REASON field will not be inserted.
ORIGINAL_ADDRESS (DSN)	Override for the “Original address:” text used in the construction of the per-recipient section of the first part of a DSN. This field should be specified in the same charset that’s used for the first part of the DSN.
REASON (DSN)	Override for the “Reason:” text used in the construction of the per-recipient section of the first part of a DSN. This field should be specified in the same charset that’s used for the first part of the DSN.
RECIPIENT_ADDRESS (DSN)	Override for the “Recipient address:” text used in the construction of the per-recipient section of the first part of a DSN. This field should be specified in the same charset that’s used for the first part of the DSN.
RETURN_PERSONAL (DSN and MDN)	Override personal name field to be used in conjunction with the From: field. This field should be RFC 2047 encoded. The value set by the RETURN_PERSONAL MTA option is used if this option is not specified.
SUBJECT (DSN and MDN)	Override Subject: field. This value will only be used if the notification didn’t provide a subject field of its own. This field should be RFC 2047 encoded. An appropriate subject will be constructed if this option isn’t used and the notification didn’t provide one.
TEXT_CHARSET (MDN)	Charset text for the first part and subject of the MDN should be converted to. The default is not to perform any conversion.

10.10.4 Additional Status Notification Message Features

The essential procedures for setting up status notification messages is describe in the previous sections. The following sections describe additional functionality:

- [“10.10.4.1 To Block Content Return on Large Messages” on page 278](#)

- “10.10.4.2 To Remove non-US-ASCII Characters from Included Headers in the Status Notification Messages” on page 278
- “10.10.4.3 To Set Notification Message Delivery Intervals” on page 278
- “10.10.4.4 To Include Altered Addresses in Status Notification Messages” on page 279
- “10.10.4.5 To Send, Block and Specify Status Notification Messages to the Postmaster” on page 280

10.10.4.1

To Block Content Return on Large Messages

Typically, when a message is bounced or blocked, the content of the message is returned to sender and to the local domain postmaster in the notification message. This can be a strain on resources if a number of very large messages are returned in their entirety. To block the return of content for messages over a certain size, set the CONTENT_RETURN_BLOCK_LIMIT option in the MTA options file.

The MTA fetches the block limit associated with the envelope return address and will set RET=HDRS if no return policy is specified and the message size exceeds the block limit. This prevents nondelivery reports for large messages from being undeliverable themselves. No new options or settings are associated with this change.

10.10.4.2

To Remove non-US-ASCII Characters from Included Headers in the Status Notification Messages

The raw format for Internet message headers does not permit non-US-ASCII characters. If non-US-ASCII characters are used in a message header they are encoded using “MIME header encoding” described in RFC 2047. Thus, a Chinese “Subject” line in an email message will actually look like this:

Subject: =?big5?Q?=A4j=AB=AC=A8=B1=AD=B1=B0=D3=F5=A5X=AF=B2?=

and it is the responsibility of email clients to remove the encoding when displaying headers.

Because the %H template copies headers into the body of the notification message, the encoded header text will normally appear. However, Messaging Server will remove the encoding if the character set in the subject (in this case “big5”) matches the character set in the Content-Type header character set parameter in `return_prefix.txt`. If it doesn’t match, the Messaging Server will leave the encoding intact.

10.10.4.3

To Set Notification Message Delivery Intervals

Keywords: notices, nonurgentnotices, normalnotices, urgentnotices

Undeliverable messages are held in a given channel queue for specified amount of time before being returned to sender. In addition, a series of status/warning messages can be returned to the sender while Messaging Server attempts delivery. The amount of time and intervals between messages can be specified with the notices, nonurgentnotices, normalnotices, or urgentnotices keywords. Examples:

```
notices 1 2 3
```

Transient failure status notification messages are sent after 1 and 2 days for all messages. If the message is still not delivered after 3 days, it is returned to its originator.

```
urgentnotices 2,4,6,8
```

Transient failure notifications are sent after 2, 4, and 6 days for messages of urgent priority. If the message is still not delivered after 8 days, it is returned to its originator.

Note that the RETURN_UNITS option in the MTA Options file allows you to specify the units in either hours (1) or days (0). The default is days (0). If you set RETURN_UNITS=1, then you will need to schedule the return job to run hourly as well to get hourly notices. When the return job runs every hour it will also roll over the mail.log* files every hour. To prevent the hourly rollover of the mail.log file, set the IMTA_RETURN_SPLIT_PERIOD tailor file option in the imta.tailor file to 24. Return job scheduling is controlled by the local.schedule.return_job configutil parameter. Note, however, that by default this command is run on a regular basis (see “[4.6.2 Pre-defined Automatic Tasks](#)” on page 130).

If no notices keyword is specified, the default is to use the notices setting for the local, l, channel. If no setting has been made for the local channel, then the default is to use notices 3, 6, 9, 12.

10.10.4.4

To Include Altered Addresses in Status Notification Messages

Keywords: includefinal, suppressfinal, useintermediate

When the MTA generates a notification message (bounce message, delivery receipt message, and so on), there may be both an “original” form of a recipient address and an altered “final” form of that recipient address available to the MTA. The MTA always includes the original form (assuming it is present) in the notification message, because that is the form that the recipient of the notification message (the sender of the original message, which the notification message concerns) is most likely to recognize.

The includefinal and suppressfinal channel keywords control whether the MTA also includes the final form of the address. Suppressing the inclusion of the final form of the address may be of interest to sites that are “hiding” their internal mailbox names from external view. Such sites may prefer that only the original, “external” form of address be included in status notification messages. includefinal is the default and includes the final form of the recipient address. suppressfinal causes the MTA to suppress the final address form, if an original address form is present, from status notification messages.

The useintermediate keyword uses an intermediate form of the address produced after list expansion, but prior to user mailbox name generation. If this information isn’t available, the final form is used.

10.10.4.5 To Send, Block and Specify Status Notification Messages to the Postmaster

By default, a copy of failure and warning status notification messages are sent to the postmaster unless error returns and warnings are completely suppressed with a blank `Errors-to:` header line or a blank envelope `From:` address. Further granularity of notification message delivery to the postmaster can be controlled by a number of channel keywords described in the sections below and in [Table 10–12](#). This section consists of the following subsections:

- “Returned Failed Messages” on page 280
- “Warning Messages” on page 280
- “Blank Envelope Return Addresses” on page 280
- “Postmaster Returned Message Content” on page 281
- “Setting Per Channel Postmaster Addresses” on page 281

Returned Failed Messages

Keywords: `sendpost`, `nosendpost`, `copysendpost`, `errsendpost`

A channel program may be unable to deliver a message because of long-term service failures or invalid addresses. When this occurs, the MTA channel program returns the message to the sender with an accompanying explanation of why the message was not delivered. Optionally, a copy of all failed messages is sent to the local postmaster. This is useful for monitoring message failures, but it can result in an excessive amount of traffic with which the postmaster must deal. (See [Table 10–12](#).)

Warning Messages

Keywords: `warnpost`, `nowarnpost`, `copywarnpost`, `errwarnpost`

In addition to returning messages, the MTA can send detailed warnings for undelivered messages. This is generally due to time-outs based on the setting of the notices channel keyword, although in some cases channel programs may produce warning messages after failed delivery attempts. The warning messages contain a description of what’s wrong and how long delivery attempts continue. In most cases they also contain the headers and the first few lines of the message in question.

Optionally, a copy of all warning messages can be sent to the local postmaster. This can be somewhat useful for monitoring the state of the various queues, although it does result in lots of traffic for the postmaster to deal with. The keywords `warnpost`, `copywarnpost`, `errwarnpost`, and `nowarnpost` are used to control the sending of warning messages to the postmaster. (See [Table 10–12](#).)

Blank Envelope Return Addresses

Keywords: `returnnenvelope`

The `returnenvelope` keyword takes a single integer value, which is interpreted as a set of bit flags. Bit 0 (value = 1) controls whether or not return notifications generated by the MTA are written with a blank envelope address or with the address of the local postmaster. Setting the bit forces the use of the local postmaster address; clearing the bit forces the use of a blank address.

Note – The use of a blank address is mandated by RFC 1123. However, some systems do not properly handle blank envelope From: addresses and may require the use of this option.

Bit 1 (value = 2) controls whether or not the MTA replaces all blank envelope addresses with the address of the local postmaster. This is used to accommodate noncompliant systems that do not conform to RFC 821, RFC 822, or RFC 1123.

Bit 2 (value = 4) prohibits syntactically invalid return addresses.

Bit 3 (value = 8) same as `mailfromdnsverify` keyword.

Postmaster Returned Message Content

Keywords:`postheadonly`, `postheadbody`

When a channel program or the periodic message return job returns messages to both the postmaster and the original sender, the postmaster copy can either be the entire message or just the headers. Restricting the postmaster copy to just the headers adds an additional level of privacy to user mail. However, this by itself does not guarantee message security; postmasters and system managers are typically in a position where the contents of messages can be read using root system privileges, if they so choose. (See [Table 10–12](#).)

Setting Per Channel Postmaster Addresses

Keywords:`aliaspostmaster`, `returnaddress`, `noreturnaddress`, `returnpersonal`, `noreturnpersonal`

By default, the Postmaster’s return address that is used when the MTA constructs bounce or status notification messages is `postmaster@local-host`, where *local-host* is the official local host name (the name on the local channel), and the Postmaster personal name is “MTA e-Mail Interconnect.” Care should be taken in the selection of the Postmaster address—an illegal selection may cause rapid message looping and a great number of error messages.

The `RETURN_ADDRESS` and `RETURN_PERSONAL` options can be used to set an MTA system default for the Postmaster address and personal name. Or, if per channel controls are desired, the `returnaddress` and `returnpersonal` channel keywords may be used. `returnaddress` and `returnpersonal` each take a required argument specifying the Postmaster address and Postmaster personal name, respectively. `noreturnaddress` and `noreturnpersonal` are the defaults and signify that the default values should be used. The defaults are established via the `RETURN_ADDRESS` and `RETURN_PERSONAL` options or the normal default values if such options are not set.

If the `aliaspostmaster` keyword is placed on a channel, then any messages addressed to the user name `postmaster` (lowercase, uppercase, or mixed case) at the official channel name is redirected to `postmaster@local-host`, where `local-host` is the official local host name (the name on the local channel). Note that Internet standards require that any domain in the DNS that accepts mail have a valid postmaster account that receives mail. So this keyword can be useful when it is desired to centralize postmaster responsibilities, rather than setting separate postmaster accounts for separate domains. That is, whereas `returnaddress` controls what return postmaster address is used when the MTA generates a notification message from the postmaster, `aliaspostmaster` affects what the MTA does with messages addressed to the postmaster.

TABLE 10–12 Keywords for Sending Notification Messages to the Postmaster and Sender

Keyword	Description
Returned Message Content	Specifies Addresses on Notifications
<code>notices</code>	Specifies the time that may elapse before notices are sent and messages returned.
<code>nonurgentnotices</code>	Specifies the time that may elapse before notices are sent and messages returned for messages of non-urgent priority.
<code>normalnotices</code>	Specifies the time that may elapse before notices are sent and messages returned for messages of normal priority.
<code>urgentnotices</code>	Specify the time which may elapse before notices are sent and messages returned for messages of urgent priority.
Returned Messages	How to handle failure notices for returned messages.
<code>sendpost</code>	Enables sending a copy of all failed messages to the postmaster.
<code>copysendpost</code>	Sends a copy of the failure notice to the postmaster unless the originator address on the failing message is blank, in which case, the postmaster gets copies of all failed messages except those messages that are actually themselves bounces or notifications.
<code>errsendpost</code>	Sends a copy of the failure notice to the postmaster only when the notice cannot be returned to the originator. If <code>nosendpost</code> is specified, failed messages are never sent to the postmaster.
<code>nosendpost</code>	Disables sending a copy of all failed messages to the postmaster.
Warning Messages	How to handle warning messages.
<code>warnpost</code>	Enables sending a copy of warning messages to the postmaster. The default is to send a copy of warnings to the postmaster unless warnings are completely suppressed with a blank <code>Warnings-to:</code> header or a blank envelope <code>From:</code> address.
<code>copywarnpost</code>	Sends a copy of the warning message to the postmaster unless the originator address on the undelivered message is blank.
<code>errwarnpost</code>	Sends a copy of the warning message to the postmaster when the notice cannot be returned to the originator.

TABLE 10-12 Keywords for Sending Notification Messages to the Postmaster and Sender *(Continued)*

Keyword	Description
nowarnpost	Disables sending a copy of warning messages to the postmaster.
Returned Message Content	Specifies whether to send entire message or just headers to the postmaster.
postheadonly	Returns only headers to the postmaster. Restricting the postmaster copy to just the headers adds an additional level of privacy to user mail. However, this does not guarantee message security as postmasters and system managers are able to read the contents of messages using root system privileges, if they choose.
postheadbody	Returns both the headers and the contents of the message.
Returned Message Content	Specifies Addresses on Notifications
includefinal	Include final form of address in delivery notifications (recipient address).
returnenvelope	Control use of blank envelope return addresses. The returnenvelope keyword takes a single integer value, which is interpreted as a set of bit flags. Bit 0 (value = 1) controls whether or not return notifications generated by the MTA are written with a blank envelope address or with the address of the local postmaster. Setting the bit forces the use of the local postmaster address; clearing the bit forces the use of a blank address. Bit 1 (value = 2) controls whether or not the MTA replaces all blank envelope addresses with the address of the local postmaster. This is used to accommodate noncompliant systems that do not conform to RFC 821, RFC 822, or RFC 1123. Bit 2 (value = 4) prohibits syntactically invalid return addresses. Bit 3 (value = 8) same as mailfromdnsverify keyword.
suppressfinal	Suppress the final address form from notification messages, if an original address form is present, from notification messages.
useintermediate	Uses an intermediate form of the address produced after list expansion, but prior to user mailbox name generation. If this information isn't available, the final form is used.
Returned Message Content	Specifies Addresses on Notifications
aliaspostmaster	Messages addressed to the user name postmaster at the official channel name is redirected to postmaster@local-host, where local-host is the local host name (the name on the local channel).
returnaddress	Specifies the return address for the local postmaster.
noreturnaddress	Use RETURN_ADDRESS option value as postmaster address name.
returnpersonal	Set the personal name for the local postmaster.
noreturnpersonal	Use RETURN_PERSONAL option value as postmaster personal name.

10.11 Controlling Message Disposition Notifications

Message Disposition Notifications (MDN) are email reports sent by the MTA to a sender and/or postmaster reporting on a message's delivery disposition. For example, if a message is rejected by a Sieve filter, an MDN will be sent to the sender. MDNs are also known as read receipts, acknowledgements, receipt notifications, or delivery receipts. The Sieve scripting language is typically used for messaging filtering and vacation messages.

10.11.1 To Customize and Localize Message Disposition Notification Messages

The instructions for modifying and localizing MDNs parallel those described in customizing and localizing delivery status notification messages with some minor differences as described here. (See “[10.10.2 To Customize and Localize Delivery Status Notification Messages](#)” on page 274 and “[10.10.3 Internationalization of Generated Notices](#)” on page 276.)

The mapping (called the `DISPOSITION_LANGUAGE` mapping) parallels the `notification_language` mapping table (see “[10.10.2 To Customize and Localize Delivery Status Notification Messages](#)” on page 274) used to internationalize status notifications.

However, probes for MDNs to this mapping take the following form:

`type|modifiers|source-channel|header-language|return|recipient`

Where:

`type` is disposition type, which can be one of the following: `displayed`, `dispatched`, `processed`, `deleted`, `denied`, or `failed`.

`modifiers` is a comma-separate list of disposition modifiers. The current list is: `error`, `warning`, `superseded`, and `expired`.

`source-channel` is the source channel producing the MDN.

`header-language` is the language specified in one of the following: `accept-language`, `preferred-language`, or `x-accept-language`. (MTA uses the first of these options that is present.)

`return` is the address to which the notification is being returned.

`recipient` is the address that the disposition is about.

The result of the disposition mapping consists of two or three pieces of information separated by vertical bars (|). The first piece of information is the directory where the template files for the disposition notification can be found. The second piece of information is the character set into

which the standalone disposition text should be forced. (This information is required because some dispositions—notably the dispositions produced by autoreply echo or the use of the :mime parameter to the vacation Sieve action—do not employ template files and consequently cannot inherit the character set from those files.) Finally, the third piece of information is an override subject line for the notification. This information is only used if the \$T flag is also set by the mapping.

The following additional template files are used to construct MDNs:

```
disposition_deleted.txt disposition_failed.txt disposition_denied.txt  
disposition_prefix.txt disposition_dispatched.txt  
disposition_processed.txt disposition_displayed.txt  
disposition_suffix.txt disposition_option.opt
```

The use of these template files parallels that of the various `return_*.txt` files for status notification messages. Message text for `*.txt` files should be limited to 78 characters per line.

10.12 Optimizing MTA Performance

This section describes miscellaneous optimizations for the MTA. It consists of the following sections:

- “[10.12.1 Optimizing Authorization Checks to the LDAP Directory for Messages Addressed to Mailing Lists](#)” on page 285

10.12.1 Optimizing Authorization Checks to the LDAP Directory for Messages Addressed to Mailing Lists

You can use metacharacter character substitutions to reduce authorization checks to the LDAP directory for messages addressed to a mailing list.

Metacharacter substitutions can now be specified in `mgrpModerator`, `mgrpAllowedBroadcaster` and `mgrpDisallowedBroadcaster` attributes. In particular, the various address-related metacharacter sequences (\$A for the entire address, \$U for the mailbox part, \$D for the domain part) refer to the current envelope `From:` address and can in some cases be used to limit the results returned by the URL to entries that are likely (or guaranteed) to match. This may make authorization checks much more efficient.

The new MTA option `PROCESS_SUBSTITUTIONS` controls whether or not substitutions are performed in various LDAP attributes that specify a URL. This is a bit-encoded value, with the bits defined as follows:

Bit	Value	Description
0	1	Enables substitutions in mgrpDisallowBroadcaster if set
1	2	Enables substitutions in mgrpAllowedBroadcaster if set
2	4	Enables substitutions in mgrpModerator if set
3	8	Enables substitutions in mgrpDeliverTo if set
4	16	Enables substitutions in memberURL

The `PROCESS_SUBSTITUTIONS` MTA option defaults to 0, meaning that all of these substitutions are disabled by default.

An example would be a dynamic list defined through an LDAP lookup where anyone on the list is allowed to post. In such cases you typically define the list with attributes like:

```
mgrpAllowedBroadcaster:  
ldap://o=Sesta,c=US??sub?(&(objectClass/inetMailUser)(objectClass/inetOrgPerson))  
mgrpDeliverTo:  
ldap://o=Sesta,c=US??sub?(&(objectClass/inetMailUser)(objectClass/inetOrgPerson))
```

The effect of such a definition, however, is to expand the list twice, once for the authorization check and once to build the actual recipient list. This is a very server intensive operation. If, on the other hand, you add a restriction so only entries containing the current envelope `From` address are returned in the authorization check, things may be much more efficient. First change the `PROCESS_SUBSTITUTION` setting to 2, and then you can set the following entries:

```
mgrpAllowedBroadcaster:  
ldap://o=Sesta,c=US??sub?(&(objectClass/inetMailUser)(objectClass/inetOrgPerson)  
(mail=$A))  
mgrpDeliverTo:  
ldap://o=Sesta,c=US??sub?(&(objectClass/inetMailUser)(objectClass/inetOrgPerson))
```

In this example, only the sender's entry is checked for broadcast authorization as opposed to all the user entries in Sesta US. This reduces the work the directory server has to do to a single (hopefully indexed) match and a single return value. The alternative is to return the entire list and have the MTA perform the match.

Note that the information available for substitution varies depending on whether the attribute is used for authorization checks or for actual list expansion. For authorization attributes, the whole address (\$A), domain (\$D), host (\$H), and local-part (\$L) are all derived from the authenticated sender address. In the case of list expansion attributes all of these substitution values are derived from the envelope recipient address that specified the list. In both cases, however, the subaddress substitution (\$S) is derived from the current envelope recipient address.

The ability to access subaddress information in list expansion URLs makes it possible to define *metagroups*, that is, a single group entry that creates an entire collection of different groups. For example, a group with a `mgrpDeliverTo` value of:

```
mgrpDeliverTo: ldap:///o=usergroup?mail?sub?(department=$S)
```

and the corresponding `PROCESS_SUBSTITUTIONS` value being 8. It is possible to send mail to every member of a given department with an address of the form `group+department@domain.com`. Note that a mechanism like a forward mapping could be used to alter the syntax if subaddresses are seen as too difficult.

Configuring Rewrite Rules

This chapter describes how to configure rewrite rules in the `imta.cnf` file. If you have not already read [Chapter 10, “About MTA Services and Configuration,”](#) you should do so before reading this chapter.

This chapter contains the following sections:

- “[11.1 Before You Begin](#)” on page 289
- “[11.2 Rewrite Rule Structure](#)” on page 290
- “[11.3 Rewrite Rule Patterns and Tags](#)” on page 291
- “[11.4 Rewrite Rule Templates](#)” on page 294
- “[11.5 How the MTA Applies Rewrite Rules to an Address](#)” on page 297
- “[11.6 Template Substitutions and Rewrite Rule Control Sequences](#)” on page 302
- “[11.7 Handling Large Numbers of Rewrite Rules](#)” on page 314
- “[11.8 Testing Rewrite Rules](#)” on page 314
- “[11.9 Rewrite Rules Example](#)” on page 315

Messaging Server’s address rewriting facility is the primary facility for manipulating and changing the host or domain portion of addresses. Messaging Server provides other facilities for address manipulation, such as aliases, the address reversal database, and specialized mapping tables. For best performance, however, rewrite rules should be used whenever possible to perform address manipulations.

11.1 Before You Begin

When you make changes to rewrite rules in the `imta.cnf` file, you must restart any programs or channels that load the configuration data only once when they start up, for example, the SMTP server, by using the `imsimta restart` command. If you are using a compiled configuration, you must recompile and then restart.

For more information about compiling configuration information and starting programs, see the [Sun Java System Messaging Server 6.3 Administration Reference](#).

11.2 Rewrite Rule Structure

Rewrite rules appear in the upper-half of the MTA configuration file, `imta.cnf`. Each rule in the configuration file appears on a single line. Comments, but not blank lines, are allowed between the rules. The rewrite rules end with a blank line, after which the channel definitions follow. The example below shows the rewrite rule section of a partial configuration file.

```
! test.cnf - An example configuration file.  
!  
! This is only an example of a configuration file. It serves  
! no useful purpose and should not be used in a real system.  
!  
a.com    $U@a-host  
b.org    $U@b-host  
c.edu    $U%c@b-daemon  
d.com    $U%d@a-daemon  
  
! Begin channel definitions
```

Rewrite rules consist of two parts: a pattern, followed by an equivalence string or *template*. The two parts must be separated by spaces, although spaces are not allowed within the parts themselves. The structure for rewrite rules is as follows:

pattern template

pattern

Indicates the string to search for in the domain name. In [Table 11–3](#) the patterns are `a.com`, `b.org`, `c.edu`, and `d.com`.

If the pattern matches the domain part of the address, the rewrite rule is applied to the address. A blank space must separate the pattern from the template. For more information about pattern syntax, see [“11.3 Rewrite Rule Patterns and Tags” on page 291](#)

template

is one of the following:

UserTemplate%DomainTemplate@ChannelTag[controls]
UserTemplate@ChannelTag[controls]
UserTemplate%DomainTemplate[controls]
UserTemplate@DomainTemplate@ChannelTag[controls]
UserTemplate@DomainTemplate@SourceRoute@ChannelTag[controls]

where

UserTemplate specifies how the user part of the address is rewritten. Substitution sequences can be used to represent parts of the original address or the results of a database lookup. The

substitution sequences are replaced with what they represent to construct the rewritten address. In [Table 11–4](#), the \$U substitution sequence is used. For more information, see [“11.6 Template Substitutions and Rewrite Rule Control Sequences” on page 302](#)

DomainTemplate specifies how the domain part of the address is rewritten. Like the *UserTemplate*, the *DomainTemplate* can contain substitution sequences.

ChannelTag indicates the channel to which this message is sent. (All channel definitions must include a channel tag as well as a channel name. The channel tag typically appears in rewrite rules, as well as in its channel definition.)

controls. The applicability of a rule can be limited using controls. Some control sequences must appear at the beginning of the rule; other controls must appear at the end of the rule. For more information about controls, see [“11.6 Template Substitutions and Rewrite Rule Control Sequences” on page 302](#)

For more information about template syntax, see [“11.4 Rewrite Rule Templates” on page 294](#)

11.3 Rewrite Rule Patterns and Tags

This section consists of the following subsections:

- [“11.3.1 A Rule to Match Percent Hacks” on page 293](#)
- [“11.3.2 A Rule to Match Bang-Style \(UUCP\) Addresses” on page 293](#)
- [“11.3.3 A Rule to Match Any Address” on page 294](#)
- [“11.3.4 Tagged Rewrite Rule Sets” on page 294](#)

Most rewrite rule patterns consist either of a specific host name that will match only that host or of a subdomain pattern that will match any host/domain in the entire subdomain.

For example, the following rewrite rule pattern contains a specific host name that will match the specified host only:

```
host.siroe.com
```

The next rewrite rule pattern contains a subdomain pattern that will match any host or domain in the entire subdomain:

```
.siroe.com
```

This pattern will not, however, match the exact host name `siroe.com`; to match the exact host name `siroe.com`, a separate `siroe.com` pattern would be needed.

The MTA attempts to rewrite host/domain names starting from the specific host name and then incrementally generalizing the name to make it less specific. This means that a more specific rewrite rule pattern will be preferentially used over more general rewrite rule patterns. For example, assume the following rewrite rule patterns are present in the configuration file:

```
hosta.subnet.siroe.com
.subnet.siroe.com
.siroe.com
```

Based on the rewrite rule patterns, an address of `jdoe@hosta.subnet.siroe.com` will match the `hosta.subnet.siroe.com` rewrite rule pattern; an address of `jdoe@hostb.subnet.siroe.com` will match the `.subnet.siroe.com` rewrite rule pattern; and an address of `jdoe@hostc.siroe.com` will match the `.siroe.com` rewrite rule pattern.

In particular, the use of rewrite rules incorporating subdomain rewrite rule patterns is common for sites on the Internet. Such a site will typically have a number of rewrite rules for their own internal hosts and subnets, and then will include rewrite rules for the top-level Internet domains into their configuration from the file `internet.rules` (*msg-svr-base/config/internet.rules*).

To ensure that messages to Internet destinations (other than to the internal host destinations handled via more specific rewrite rules) will be properly rewritten and routed to an outgoing TCP/IP channel, ensure that the `imta.cnf` file contains:

- Rewrite rules with patterns that match the top level Internet domains
- Templates that rewrite addresses matching such patterns to an outgoing TCP/IP channel

```
!      Ascension Island
.AC          $U%$H$D@TCP-DAEMON
. [text
.     removed for
.     brevity]
!      Zimbabwe
.ZW          $U%$H$D@TCP-DAEMON
```

IP domain literals follow a similar hierarchical matching pattern, though with right-to-left (rather than left-to-right) matching. For example, the following pattern matches only and exactly the IP literal `[1.2.3.4]`:

`[1.2.3.4]`

The next pattern matches anything in the `1.2.3.0` subnet:

`[1.2.3.]`

In addition to the more common sorts of host or subdomain rewrite rule patterns already discussed, rewrite rules may also make use of several special patterns, summarized in [Table 11–1](#), and discussed in the following subsections.

TABLE 11-1 Summary of Special Patterns for Rewrite Rules

Pattern	Description/Usage
\$*	Matches any address. This rule, if specified, is tried first regardless of its position in the file.
\$%	Percent Hack Rule. Matches any host/domain specification of the form A%B.
\$!	Bang-style Rule. Matches any host/domain specification of the form B!A.
[]	IP literal match-all rule. Matches any IP domain literal.
.	Matches any host/domain specification. For example, joe@[129.165.12.11]

In addition to these special patterns, Messaging Server also has the concept of *tags*, which may appear in rewrite rule patterns. These tags are used in situations where an address may be rewritten several times and, based upon previous rewrites, distinctions must be made in subsequent rewrites by controlling which rewrite rules match the address. For more information, see “[11.3.4 Tagged Rewrite Rule Sets](#)” on page 294

11.3.1 A Rule to Match Percent Hacks

If the MTA tries to rewrite an address of the form A%B and fails, it tries one extra rule before falling through and treating this address form as A%B@localhost. (For more information about these address forms, see “[11.4 Rewrite Rule Templates](#)” on page 294.) This rule is only activated when a local part containing a percent sign has failed to rewrite any other way (including the match all rule described below).

The percent hack rule is useful for assigning some special, internal meaning to percent hack addresses.

11.3.2 A Rule to Match Bang-Style (UUCP) Addresses

If the MTA tries to rewrite an address of the form B!A and fails, it tries one extra rule before falling through and treating this address form as B!A@localhost. This extra rule is the *bang-style rule*. The pattern is \$!. The pattern never changes. This rule is only activated when a local part containing an exclamation point has failed to rewrite any other way (including the default rule described below).

The bang-style rule can be used to force UUCP style addresses to be routed to a system with comprehensive knowledge of UUCP systems and routing.

11.3.3 A Rule to Match Any Address

The special pattern “.” (a single period) will match any host/domain specification if no other rule matches and the host/domain specification cannot be found anywhere in the channel table. In other words, the “.” rule is used as a last resort when address rewriting would fail otherwise.

Note – Regarding substitution sequences, when the match-all rule matches and its template is expanded, \$H expands to the full host name and \$D expands to a single dot “.”. Thus, \$D is of limited use in a match-all rule template!

11.3.4 Tagged Rewrite Rule Sets

As the rewrite process proceeds it may be appropriate to bring different sets of rules into play. This is accomplished by the use of the rewrite rule tag. The current tag is prepended to each pattern before looking it up in the configuration file or domain database. The tag can be changed by any rewrite rule that matches by using the \$T substitution string in the rewrite rule template (described below).

Tags are somewhat sticky; once set they will continue to apply to all hosts that are extracted from a single address. This means that care must be taken to provide alternate rules that begin with the proper tag values once any tags are used. In practice this is rarely a problem since tags are usually used in only very specialized applications. Once the rewriting of the address is finished the tag is reset to the default tag—an empty string.

By convention all tag values end in a vertical bar |. This character is not used in normal addresses and thus is free to delineate tags from the rest of the pattern.

11.4 Rewrite Rule Templates

The following sections describe in more detail template formats for rewrite rules. [Table 11–2](#) summarizes the template formats.

TABLE 11–2 Summary of Template Formats for Rewrite Rules

Template	Usage
A%B	A becomes the new user/mailbox name, B becomes the new host/domain specification, rewrite again. “11.4.2 Repeated Rewrites Template, A%B” on page 295
A@B	Treated as A%B@B. “11.4.1 Ordinary Rewriting Templates, A%B@C or A@B” on page 295

TABLE 11–2 Summary of Template Formats for Rewrite Rules *(Continued)*

Template	Usage
A%B@C	A becomes the new user/mailbox name, B becomes the new host/domain specification, route to the channel associated with the host C. “11.4.1 Ordinary Rewriting Templates, A%B@C or A@B” on page 295
A@B@C	Treated as A@B@C@C. “11.4.3 Specified Route Rewriting Templates, A@B@C@D or A@B@C” on page 296
A@B@C@D	A becomes the new user/mailbox name, B becomes the new host/domain specification, insert C as a source route, route to the channel associated with the host D. “11.4.3 Specified Route Rewriting Templates, A@B@C@D or A@B@C” on page 296

11.4.1 Ordinary Rewriting Templates, A%B@C or A@B

The following template is the most common form of template. The rule is applied to the user part of the address and to the domain part of the address. The new address is then used to route the message to a specific channel (indicated by *ChannelTag*).

UserTemplate%DomainTemplate@ChannelTag[controls]

The next form of template is identical in application to the most common form of template. However, this form of template is possible only if *DomainTemplate* and *ChannelTag* are identical.

UserTemplate@ChannelTag[controls]

11.4.2 Repeated Rewrites Template, A%B

The following template format is used for meta-rules that require additional rewriting after application of the rule. After the rule is applied, the entire rewriting process is repeated on the resulting new address. (All other rewrite rule formats cause the rewriting process to terminate after the rule has been applied.)

UserTemplate%DomainTemplate[controls]

Note, however, that while the special A%B form does cause rewriting of the current domain to restart, it is actually just a continuation of the current rewriting process. It does not rewrite the entire process from the beginning. It does not perform the \$* pattern when it goes through the second time.

For example, the following rule has the effect of removing all occurrences of the . removable domain from the ends of addresses:

```
. removable $U%$H
```

Extreme care must be taken when using these repeating rules; careless use can create a “rules loop.” For this reason meta-rules should only be used when absolutely necessary. Be sure to test meta-rules with the `imsimta test -rewrite` command. For more information on the `test -rewrite` command, see the [Sun Java System Messaging Server 6.3 Administration Reference](#).

11.4.3 Specified Route Rewriting Templates, A@B@C@D or A@B@C

The following template format works in the same way as the more common template `UserTemplate%DomainTemplate@ChannelTag` (note the difference in the first separator character), except that `ChannelTag` is inserted into the address as a source route. The message is then routed to `ChannelTag`:

`UserTemplate@DomainTemplate@Source-Route
@ChannelTag[controls]`

The rewritten address becomes `@route:user@domain`. The following template is also valid:

`UserTemplate@DomainTemplate@ChannelTag[controls]`

For example, the following rule rewrites the address `jdoe@com1` into the source-routed address `@siroe.com:jdoe@com1`. The channel tag becomes `siroe.com`:

`com1 $U@com1@siroe.com`

11.4.4 Case Sensitivity in Rewrite Rule Templates

Unlike the patterns in rewrite rules, character case in templates is preserved. This is necessary when using rewrite rules to provide an interface to a mail system that is sensitive to character case. Note that substitution sequences like `$U` and `$D` that substitute material extracted from addresses also preserve the original case of characters.

When it is desirable to force substituted material to use a particular case, for example, to force mailboxes to lowercase on UNIX systems, special substitution sequences can be used in templates to force substituted material to the desired case. Specifically, `$\` forces subsequent substituted material into lower case, `$^` forces subsequent substituted material into upper case, and `$_` says to use the original case.

For example, you can use the following rule to force mailboxes to lowercase for `unix.siroe.com` addresses:

`unix.siroe.com $\\$U$_%unix.siroe.com`

11.5 How the MTA Applies Rewrite Rules to an Address

The following steps describe how the MTA applies rewrite rules to a given address:

1. The MTA extracts the first host or domain specification from an address.
An address can specify more than one host or domain name as in the case:
`jdoe%hostname@siroe.com.`
2. After identifying the first host or domain name, the MTA conducts a search that scans for a rewrite rule whose pattern matches the host or domain name.
3. When the matching rewrite rule is found, the MTA rewrites the address according to the template portion of that rule.
4. Finally, the MTA compares the channel tag with the host names that are associated with each channel.

If a match is found, the MTA enqueues the message to the associated channel; otherwise, the rewrite process fails. If the matching channel is the local channel, some additional rewriting of the address may take place by looking up the alias database and alias file.

These steps are described in more detail in the subsections that follow.

Note – Using a channel tag that does not belong to any existing channel will cause messages whose addresses match this rule to be bounced. That is, it makes the matching messages nonroutable.

This section consists of the following subsections:

- “[11.5.1 Step 1. Extract the First Host or Domain Specification](#)” on page 297
- “[11.5.2 Step 2. Scan the Rewrite Rules](#)” on page 299
- “[11.5.3 Step 3. Rewrite Address According to Template](#)” on page 300
- “[11.5.4 Step 4. Finish the Rewrite Process](#)” on page 300
- “[11.5.5 Rewrite Rule Failure](#)” on page 301
- “[11.5.6 Syntax Checks After Rewrite](#)” on page 301
- “[11.5.7 Handling Domain Literals](#)” on page 301

11.5.1 Step 1. Extract the First Host or Domain Specification

The process of rewriting an address starts by extracting the first host or domain specification from the address. (Readers not familiar with RFC 822 address conventions are advised to read that standard to understand the following discussion.) The order in which host/domain specifications in the address are scanned is as follows:

1. Hosts in source routes (read from left to right)
2. Hosts appearing to the right of the “at” sign (@)

3. Hosts appearing to the right of the last single percent sign (%)
4. Hosts appearing to the left of the first exclamation point (!)

The order of the last two items is switched if the `bangoverpercent` keyword is in effect on the channel that is doing the address rewriting. That is, if the channel attempting to enqueue the message is, itself, marked with the `bangoverpercent` channel keyword.

Some examples of addresses and the host names that could be extracted first are shown in [Table 11–3](#).

TABLE 11–3 Extracted Addresses and Host Names

Address	First Host Domain Specification	Comments
user@a	a	A “short-form” domain name.
user@a.b.c	a.b.c	A “fully qualified” domain name (FQDN).
user@[0.1.2.3]	[0.1.2.3]	A “domain literal.”
@a:user@b.c.d	a	Source-routed address with a short-form domain name, the “route.”
@a.b.c:user@d.e.f	a.b.c	Source-routed address; route part is fully qualified.
@[0.1.2.3]:user@d.e.f	[0.1.2.3]	Source-routed address; route part is a domain literal.
@a,@b,@c:user@d.e.f	a	Source-routed address with an a to b to c routing.
@a,@[0.1.2.3]:user@b	a	Source-routed address with a domain literal in the route part.
user%A@B	B	This nonstandard form of routing is called a “percent hack.”
user%A	A	
user%A%B	B	
user%%A%B	B	
A!user	A	“Bang-style” addressing; commonly used for UUCP.
A!user@B	B	
A!user%B@C	C	
A!user%B	B	nobangoverpercent keyword active; the default.
A!user% B	A	bangoverpercent keyword active.

RFC 822 does not address the interpretation of exclamation points (!) and percent signs (%) in addresses. Percent signs are customarily interpreted in the same manner as at signs (@) if no at sign is present, so this convention is adopted by the Messaging Server MTA.

The special interpretation of repeated percent signs is used to allow percent signs as part of local user names; this might be useful in handling some foreign mail system addresses. The interpretation of exclamation points conforms to RFC 976's "bang-style" address conventions and makes it possible to use UUCP addresses with the Messaging Server MTA.

The order of these interpretations is not specified by either RFC 822 or RFC 976, so the bangoverpercent and nobangoverpercent keywords can be used to control the order in which they are applied by the channel doing the rewriting. The default is more "standard," although the alternate setting may be useful under some circumstances.

Note – The use of exclamation points (!) or percent signs (%) in addresses is not recommended.

11.5.2

Step 2. Scan the Rewrite Rules

Once the first host or domain specification has been extracted from the address, the MTA consults the rewrite rules to find out what to do with it. The host/domain specification is compared with the pattern part of each rule (that is, the left side of each rule). The comparison is case insensitive. Case insensitivity is mandated by RFC 822. The MTA is insensitive to case but preserves it whenever possible.

If the host or domain specification does not match any pattern, in which case it is said to "not match any rule," the first part of the host or domain specification—the part before the first period, usually the host name—is removed and replaced with an asterisk (*) and another attempt is made to locate the resulting host or domain specification, but only in the configuration file rewrite rules (the domain database is not consulted).

If this fails, the first part is removed and the process is repeated. If this also fails the next part is removed (usually a subdomain) and the rewriter tries again, first with asterisks and then without. All probes that contain asterisks are done only in the configuration file rewrite rules table; the domain database is not checked. This process proceeds until either a match is found or the entire host or domain specification is exhausted. The effect of this procedure is to try to match the most specific domain first, working outward to less specific and more general domains.

A more algorithmic view of this matching procedure is:

- The host/domain specification is used as the initial value for the comparison strings spec_1 and spec_2. (For example, spec_1 = spec_2 = a.b.c).
- The comparison string spec_1 is compared with the pattern part of each rewrite rule in the configuration file and then the domain database until a match is found. The matching procedure is exited if a match is found.

- If no match is found, then the left-most, nonasterisk part of `spec_2` is converted to an asterisk. For example, if `spec_2` is `a.b.c` then it is changed to `*.b.c`; if `spec_2` is `*.b.c`, then it is changed to `*.*.c`. The matching procedure is exited if a match is found.
- If no match is found then the first part, including any leading period, of the comparison string `spec_1` is removed. Where `spec_1` has only one part (for example, `.c` or `c`), the string is replaced with a single period, “.”. If the resulting string `spec_1` is of nonzero length, then you return to step 1. If the resulting string has zero length (for example, was previously “”), then the lookup process has failed and you exit the matching procedure.

For example, suppose the address `dan@sc.cs.siroe.edu` is to be rewritten. This causes the MTA to look for the following patterns in the given order:

```
sc.cs.siroe.edu  
*.cs.siroe.edu  
.cs.siroe.edu  
*.*.siroe.edu  
.siroe.edu  
*.*.*.edu  
.edu  
*.*.*.*  
.
```

11.5.3 Step 3. Rewrite Address According to Template

Once the host/domain specification matches a rewrite rule, it is rewritten using the template part of the rule. The template specifies three things:

1. A new user name for the address.
2. A new host/domain specification for the address.
3. A channel tag that identifies an existing MTA channel to which messages to this address should be sent.

11.5.4 Step 4. Finish the Rewrite Process

One of two things can happen once the host/domain specification is rewritten.

- If the channel tag is associated neither with the local channel nor a channel marked with the `routelocal` channel keyword, or there are no additional host/domain specifications in the address, the rewritten specification is substituted into the address replacing the original specification that was extracted for rewriting, and the rewriting process terminates.

- If the channel tag matches the local channel or a channel marked `routelocal` and there are additional host/domain specifications that appear in the address, the rewritten address is discarded, the original (initial) host/domain specification is removed from the address, a new host/domain specification is extracted from the address, and the entire process is repeated. Rewriting will continue until either all the host/domain specifications are gone or a route through a non-local, non-`routelocal` channel is found. This iterative mechanism is how the MTA provides support for source routing. In effect, superfluous routes through the local system and `routelocal` systems are removed from addresses by this process.

11.5.5 Rewrite Rule Failure

If a host/domain specification fails to match any rewrite rule and no default rule is present, the MTA uses the specification “as-is”; for example, the original specification becomes both the new specification and the routing system. If the address has a nonsensical host/domain specification it will be detected when the routing system does not match any system name associated with any channel and the message will be bounced.

11.5.6 Syntax Checks After Rewrite

No additional syntax checking is done after the rewrite rules have been applied to an address. This is deliberate—it makes it possible for rewrite rules to be used to convert addresses into formats that do not conform to RFC 822. However, this also means that mistakes in the configuration file may result in messages leaving the MTA with incorrect or illegal addresses.

11.5.7 Handling Domain Literals

Domain literals are handled specially during the rewriting process. If a domain literal appearing in the domain portion of an address does not match a rewrite rule pattern as is, the literal is interpreted as a group of strings separated by periods and surrounded by square brackets. The right-most string is removed and the search is repeated. If this does not work, the next string is removed, and so on until only empty brackets are left. If the search for empty brackets fails, the entire domain literal is removed and rewriting proceeds with the next section of the domain address, if there is one. No asterisks are used in the internal processing of domain literals; when an entire domain literal is replaced by an asterisk, the number of asterisks corresponds to the number of elements in the domain literal.

Like normal domain or host specifications, domain literals are also tried in most specific to least specific order. The first rule whose pattern matches will be the one used to rewrite the host or domain specification. If there are two identical patterns in the rules list, the one which appears first will be used.

As an example, suppose the address `dan@[128.6.3.40]` is to be rewritten. The rewriter looks for `[128.6.3.40]`, then `[128.6.3.]`, then `[128.6.]`, then `[128.]`, then `[]`, then `[*.*.*.*]`, and finally the match-all rule “`.`”.

11.6 Template Substitutions and Rewrite Rule Control Sequences

Substitutions are used to rewrite user names or addresses by inserting a character string into the rewritten address, the value of which is determined by the particular substitution sequence used. This section consists of the following subsections:

- “11.6.1 Username and Subaddress Substitution, `$U, $0U, $1U`” on page 305
- “11.6.2 Host/Domain and IP Literal Substitutions, `$D, $H, $nD, $nH, $L`” on page 305
- “11.6.3 Literal Character Substitutions, `$$, $$%, $$@`” on page 306
- “11.6.4 LDAP Query URL Substitutions, `$]...[`” on page 306
- “11.6.5 General Database Substitutions, `$(...)`” on page 307
- “11.6.6 Apply Specified Mapping, `$$...{}`” on page 308
- “11.6.7 Customer-supplied Routine Substitutions, `$[...]`” on page 308
- “11.6.8 Single Field Substitutions, `$&, $!, $*, $#`” on page 309
- “11.6.9 Unique String Substitutions” on page 310
- “11.6.10 Source-Channel-Specific Rewrite Rules (`$M, $N`)” on page 310
- “11.6.11 Destination-Channel-Specific Rewrite Rules (`$C, $Q`)” on page 310
- “11.6.12 Direction-and-Location-Specific Rewrite Rules (`$B, $E, $F, $R`)” on page 311
- “11.6.13 Host-Location-Specific Rewrites (`$A, $P, $S, $X`)” on page 312
- “11.6.14 Changing the Current Tag Value, `$T`” on page 312
- “11.6.15 Controlling Error Messages Associated with Rewriting (`$?`)” on page 313

For example, in the following template, the `$U` is a substitution sequence. It causes the *username* portion of the address being rewritten to be substituted into the output of the template. Thus, if `jdoe@mailhost.siroe.com` was being rewritten by this template, the resulting output would be `jdoe@siroe.com`, the `$U` substituting in the *username* portion, `jdoe`, of the original address:

```
$U@siroe.com
```

Control sequences impose additional conditions to the applicability of a given rewrite rule. Not only must the pattern portion of the rewrite rule match the host or domain specification being examined, but other aspects of the address being rewritten must meet conditions set by the control sequence or sequences. For example, the `$E` control sequence requires that the address being rewritten be an envelope address, while the `$F` control sequence requires that it be a forward pointing address. The following rewrite rule only applies to (rewrite) envelope To: addresses of the form `user@siroe.com`:

```
siroe.com $U@mail.siroe.com$E$F
```

If a domain or host specification matches the pattern portion of a rewrite rule but doesn't meet all of the criteria imposed by a control sequences in the rule's template, then the rewrite rule fails and the rewriter continues to look for other applicable rules.

Table 11–4 summarizes the template substitutions and control sequences.

TABLE 11–4 Summary of Rewrite Rule Template Substitutions and Control Sequences

Substitution Sequence	Substitutes
\$D	Portion of domain specification that matched.
\$H	Unmatched portion of host/domain specification; left of dot in pattern.
\$L	Unmatched portion of domain literal; right of dot in pattern literal.
\$U	User name from original address.
\$nA	Inserts the nth left character of the current address starting from position 0. The entire address is inserted if n is omitted.
\$nX	Inserts the nth left component of the mailhost starting from 0. The entire mailhost is inserted if n is omitted.
\$0U	Local part (username) from original address, minus any subaddress.
\$1U	Subaddress, if any, from local part (username) of original address.
\$\$	Inserts a literal dollar sign (\$).
\$%	Inserts a literal percent sign (%).
\$@	Inserts a literal at sign (@).
\$\	Forces material to lowercase.
\$^	Forces material to uppercase.
\$_	Uses original case.
\$=	Forces subsequent substituted characters to undergo quoting appropriate for insertion into LDAP search filters.
\$W	Substitutes in a random, unique string.
\$]...[LDAP search URL lookup.
\$. .	Establish a string which will be processed as the mapping entry result in the event of a temporary LDAP lookup failure.
\$(text)	General database substitution; rule fails if lookup fails.
\${...}	Applies specified mapping to supplied string.
\$[...]	Invoke customer supplied routine; substitute in result.

TABLE 11-4 Summary of Rewrite Rule Template Substitutions and Control Sequences *(Continued)*

Substitution Sequence	Substitutes
\$&n	The <i>n</i> th part of unmatched (or wildcarded) host, counting from left to right, starting from 0.
\$!n	The <i>n</i> th part of unmatched (or wildcarded) host, as counted from right to left, starting from 0.
\$*n	The <i>n</i> th part of matching pattern, counting from left to right, starting from 0.
\$#n	The <i>n</i> th part of matching pattern, counted from right to left, starting from 0.
\$nD	Portion of domain specification that matched, preserving from the <i>n</i> th leftmost part starting from 0
\$nH	Portion of host/domain specification that didn't match, preserving from the <i>n</i> th leftmost part starting from 0
Control Sequence	Effect on Rewrite Rule
\$1M	Apply only if the channel is an internal reprocessing channel.
\$1N	Apply only if the channel is not an internal reprocessing channel.
\$1~	Perform any pending channel match checks. If the checks fail, successfully terminate processing of the current rewrite rule template.
\$A	Apply if host is to the right of the at sign
\$B	Apply only to header/body addresses
\$C <i>channel</i>	Fail if sending to <i>channel</i>
\$E	Apply only to envelope addresses
\$F	Apply only to forward-directed (e.g., To:) addresses
\$M <i>channel</i>	Apply only if <i>channel</i> is rewriting the address
\$N <i>channel</i>	Fail if <i>channel</i> is rewriting the address
\$P	Apply if host is to the right of a percent sign
\$Q <i>channel</i>	Apply if sending to <i>channel</i>
\$R	Apply only to backwards-directed (e.g., From:) addresses
\$S	Apply if host is from a source route
\$Tnewtag	Set the rewrite rule tag to newtag
\$Vhost	Fail if the host name is not defined in the LDAP directory (either in the DC tree or as a virtual domain). If the LDAP search times out, the remainder of the rewrite pattern from directly after the character following the host name is replaced with the MTA option string DOMAIN_FAILURE.
\$X	Apply if host is to the left of an exclamation point
\$Zhost	Fail if the host name is defined in the LDAP directory (either in the DC tree or as a virtual domain). If the LDAP search times out, the remainder of the rewrite pattern from directly after the character following the host name is replaced with the MTA option string DOMAIN_FAILURE.

TABLE 11-4 Summary of Rewrite Rule Template Substitutions and Control Sequences *(Continued)*

Substitution Sequence	Substitutes
\$nT	Overrides the default ALIAS_MAGIC setting, where <i>n</i> is an appropriate value for the ALIAS_MAGIC MTA option. Overrides the setting for the domain when the rule matches during alias expansion.
\$?errmsg	If rewriting fails, return <i>errmsg</i> instead of the default error message. The error message must be in US ASCII.
\$number?errmsg	If rewriting fails, return <i>errmsg</i> instead of the default error message, and set the SMTP extended error code to <i>a.b.c</i> : <ul style="list-style-type: none"> ■ <i>a</i> is <i>number</i>/ 1000000 (the first digit) ■ <i>b</i> is (<i>number</i>/1000) remainder 1000 (the value of the digits 2 through 4) ■ <i>c</i> is <i>number</i> remainder 1000 (the value of the last three digits). The following example sets the error code to 3.45.89: <code>\$3045089?the snark is a boojum</code>

11.6.1 Username and Subaddress Substitution, \$U, \$0U, \$1U

Any occurrences of \$U in the template are replaced with the username (RFC 822 “local-part”) from the original address. Note that user names of the form a.“b” will be replaced by “a.b” as RFC2822 deprecates the former syntax from RFC 822 and it is expected that the latter usage will become mandatory in the future.

Any occurrences of \$0U in the template are replaced with the username from the original address, minus any subaddress and subaddress indication character (+). Any occurrences of \$1U in the template are replaced with the subaddress and subaddress indication character, if any, from the original address. So note that \$0U and \$1U are complementary pieces of the username, with \$0U\$1U being equivalent to a simple \$U.

11.6.2 Host/Domain and IP Literal Substitutions, \$D, \$H, \$nD, \$nH, \$L

Any occurrences of \$H are replaced with the portion of the host/domain specification that was not matched by the rule. Any occurrences of \$D are replaced by the portion of the host/domain specification that was matched by the rewrite rule. The \$nH and \$nD characters are variants that preserve the normal \$H or \$D portion from the nth leftmost part starting counting from 0. That is, \$nH and \$nD omit the leftmost n parts (starting counting from 1) of what would normally be a \$H or \$D, substitution, respectively. In particular, \$0H is equivalent to \$H and \$0D is equivalent to \$D.

For example, assume the address jdoe@host.siroe.com matches the following rewrite rule:

```
host.siroe.com    $U%$1D@TCP-DAEMON
```

The resulting address is `jdoe@siroe.com` with `TCP-DAEMON` used as the outgoing channel. Here where `$D` would have substituted in the entire domain that matched, `host.siroe.com`, the `$1D` instead substitutes in the portions of the match starting from part 1 (part 1 being `siroe`), so substitutes in `siroe.com`.

`$L` substitutes the portion of a domain literal that was not matched by the rewrite rule.

11.6.3 Literal Character Substitutions, \$\$, \$%, \$@

The `$`, `%`, and `@` characters are normally metacharacters in rewrite rule templates. To perform a literal insertion of such a character, quote it with a dollar character, `$`. That is, `$$` expands to a single dollar sign, `$`; `$%` expands to a single percent, `%` (the percent is not interpreted as a template field separator in this case); and `$@` expands to a single at sign, `@` (also not interpreted as a field separator).

11.6.4 LDAP Query URL Substitutions, \$]...[

A substitution of the form `$]ldap-url[` is interpreted as an LDAP query URL and the result of the LDAP query is substituted. Standard LDAP URLs are used with the host and port omitted. The host and port are instead specified in the `msg.conf` file (`local.ldaphost` and `local.ldapport` attributes).

That is, the LDAP URL should be specified as follows where the square bracket characters, `[]`, indicate optional portions of the URL:

```
ldap:///dn[?attributes[?scope?filter]]
```

The `dn` is required and is a distinguished name specifying the search base. The optional attributes, scope, and filter portions of the URL further refine what information to return. For a rewrite rule, the desired attributes to specify returning might be a `mailRoutingSystem` attribute (or some similar attribute). The scope may be any of `base` (the default), `one`, or `sub`. And the desired filter might be to request the return of the object whose `mailDomain` value matches the domain being rewritten.

If the LDAP directory schema includes attributes `mailRoutingSystem` and `mailDomain`, then a possible rewrite rule to determine to which system to route a given sort of address might appear as the following where here the LDAP URL substitution sequence `$D` is used to substitute in the current domain name into the LDAP query constructed:

```
.siroe.com \
$U%$H$D@$]ldap:///o=siroe.com?mailRoutingSystem?sub? \
(mailDomain=$D)
```

For ease in reading, the backslash character is used to continue the single logical rewrite rule line onto a second physical line. [Table 11–5](#) lists the LDAP URL Substitution Sequences.

TABLE 11-5 LDAP URL Substitution Sequences

Substitution Sequence	Description
\$\$	Literal \$ character
\$..	Establishes a string which will be processed as the mapping entry result in the event of a temporary LDAP lookup failure. By default a temporary failure string remains set only for the duration of the current rule. "\$. ." can be used to return to the default state where no temporary failure string is set and temporary LDAP failures cause mapping entry or rewrite rule failure. Note that all errors other than failure to match an entry in the directory are considered to be temporary errors; in general it isn't possible to distinguish between errors caused by incorrect LDAP URLs and errors caused by directory server configuration problems.
\$~ account	Home directory of user account
\$A	Address
\$D	Domain name
\$H	Host name (first portion of fully qualified domain name)
\$L	Username minus any special leading characters such as ~ or _
\$S	Subaddress
\$U	Username

The MTA now caches URL results from lookups done in rewrite rules and mappings. This new URL result cache is controlled by two new MTA options, `URL_RESULT_CACHE_SIZE` (default 10000 entries) and `URL_RESULT_CACHE_TIMEOUT` (default 600 seconds).

11.6.5

General Database Substitutions, \$(...)

A substitution of the form `$(text)` is handled specially. The text part is used as a key to access the special general text database. This database consists of the file specified with the `IMTA_GENERAL_DATABASE` option in the `msg-svr-base/config/imta_tailor` file, which is usually the file `msg-svr-base/db/generaldb.db`.

If “text-string” is found in the database, the corresponding template from the database is substituted. If “text-string” does not match an entry in the database, the rewrite process fails; it is as if the rewrite rule never matched in the first place. If the substitution is successful, the template extracted from the database is re-scanned for additional substitutions. However, additional `$(text)` substitutions from the extracted template are prohibited in order to prevent endless recursive references.

As an example, suppose that the address `jdoe@siroe.siroen.net` matches the following rewrite rule:

```
.SIROENET $($H)
```

Then, the text string `siroe` will be looked up in the general database and the result of the look up, if any, is used for the rewrite rule's template. Suppose that the result of looking up `siroe` is `$u%eng.siroe.com@siroenet`. Then the output of the template will be `jdoe@eng.siroe.com` (i.e., `username = jdoe`, `host/domain specification = eng.siroe.com`), and the routing system will be `siroenet`.

If a general text database exists it should be world readable to insure that it operates properly. See “[10.9.1 MTA Text Databases](#)” on page 265 for more information.

11.6.6 Apply Specified Mapping, \${...}

A substitution of the form `.SIROENET $($H) ${mapping,argument}` is used to find and apply a mapping from the MTA mapping file. The `mapping` field specifies the name of the mapping table to use while `argument` specifies the string to pass to the mapping. The mapping must exist and must set the `$Y` flag in its output if it is successful; if it doesn't exist or doesn't set `$Y` the rewrite will fail. If successful the result of the mapping is merged into the template at the current location and re-expanded.

This mechanism allows the MTA rewriting process to be extended in various complex ways. For example, the `username` part of an address can be selectively analyzed and modified, which normally isn't a feature the MTA rewriting process is capable of.

11.6.7 Customer-supplied Routine Substitutions, \$[...]

A substitution of the form `$[image,routine,argument]` is used to find and call a customer-supplied routine. At run-time on UNIX, the MTA uses `dlopen` and `dlSym` to dynamically load and call the specified routine from the shared library image. The routine is then called as a function with the following argument list:

```
status := routine (argument, arglength, result, reslength)
```

`argument` and `result` are 252 byte long character string buffers. On UNIX, `argument` and `result` are passed as a pointer to a character string, (for example, in C, as `char*`.) `arglength` and `reslength` are signed, long integers passed by reference. On input, `argument` contains the argument string from the rewrite rule template, and `arglength` the length of that string. On return, the resultant string should be placed in `result` and its length in `reslength`. This resultant string will then replace the “[`image,routine,argument`]” in the rewrite rule template. The routine should return 0 if the rewrite rule should fail and -1 if the rewrite rule should succeed.

This mechanism allows the rewriting process to be extended in all sorts of complex ways. For example, a call to some type of name service could be performed and the result used to alter the address in some fashion. Directory service lookups for forward pointing addresses (e.g., To:

addresses) to the host `siroe.com` might be performed as follows with the following rewrite rule. The `$F`, described in “[11.6.12 Direction-and-Location-Specific Rewrite Rules \(\\$B, \\$E, \\$F, \\$R\)](#)” on page 311 causes this rule to be used only for forward pointing addresses:

```
siroe.com $F$[LOOKUP_IMAGE,LOOKUP,$U]
```

A forward pointing address `j doe@siroe.com` will, when it matches this rewrite rule, cause `LOOKUP_IMAGE` (which is a shared library on UNIX) to be loaded into memory, and then cause the routine `LOOKUP` called with `j doe` as the argument parameter. The routine `LOOKUP` might then return a different address, say, `John.Doe%eng.siroe.com` in the result parameter and the value `-1` to indicate that the rewrite rule succeeded. The percent sign in the result string (see “[11.4.2 Repeated Rewrites Template, A%B](#)” on page 295) `John.Doe@eng.siroe.com` as the address to be rewritten.

On UNIX systems, the site-supplied shared library image should be world readable.

11.6.8 Single Field Substitutions, \$&,amp;,\$!,&\$*,&\$#

Single field substitutions extract a single subdomain part from the host/domain specification being rewritten. The available single field substitutions are shown in [Table 11–6](#).

TABLE 11–6 Single Field Substitutions

Control Sequence	Usage
<code>\$&n</code>	Substitute the nth element, n=0,1,2,...,9, in the host specification (the part that did not match or matched a wildcard of some kind). Elements are separated by dots; the first element on the left is element zero. The rewrite fails if the requested element does not exist.
<code>\$!n</code>	Substitute the nth element, n=0,1,2,...,9, in the host specification (the part that did not match or matched a wildcard of some kind). Elements are separated by dots; the first element on the right is element zero. The rewrite fails if the requested element does not exist.
<code>\$*n</code>	Substitute the nth element, n=0,1,2,...,9, in the domain specification (the part that did match explicit text in the pattern). Elements are separated by dots; the first element on the left is element zero. The rewrite fails if the requested element does not exist.
<code>\$#n</code>	Substitute the nth element, n=0,1,2,...,9, in the domain specification (the part that did match explicit text in the pattern). Elements are separated by dots; the first element on the right is element zero. The rewrite fails if the requested element does not exist.

Suppose the address `j doe@eng.siroe.com` matches the following rewrite rule:

```
*.SIROE.COM      $U$&0.siroe.com@mailhub.siroe.com
```

Then the result from the template will be `jdoe@eng.siroe.com` with `mailhub.siroe.com` used as the routing system.

11.6.9 Unique String Substitutions

Each use of the `$W` control sequence inserts a text string composed of upper case letters and numbers that is designed to be unique and not repeatable. `$W` is useful in situation where non-repeating address information must be constructed.

11.6.10 Source-Channel-Specific Rewrite Rules (`$M`, `$N`)

It is possible to have rewrite rules that act only in conjunction with specific source channels. This is useful when a short-form name has two meanings:

1. When it appears in a message arriving on one channel.
2. When it appears in a message arriving on a different channel.

Source-channel-specific rewriting is associated with the channel program in use and the channel keywords `rules` and `norules`. If `norules` is specified on the channel associated with an MTA component that is doing the rewriting, no channel-specific rewrite checking is done. If `rules` is specified on the channel, then channel-specific rule checks are enforced. The keyword `rules` is the default.

Source-channel-specific rewriting is not associated with the channel that matches a given address. It depends only on the MTA component doing the rewriting and that component's channel table entry.

Channel-specific rewrite checking is triggered by the presence of a `$N` or `$M` control sequence in the template part of a rule. The characters following the `$N` or `$M`, up until either an at sign (@), percent sign (%), or subsequent `$N`, `$M`, `$Q`, `$C`, `$T`, or `$?` are interpreted as a channel name.

For example, `$Mchannel` causes the rule to fail if *channel* is not currently doing the rewriting. `$Nchannel` causes the rule to fail if *channel* is doing the rewriting. Multiple `$M` and `$N` clauses may be specified. If any one of multiple `$M` clauses matches, the rule succeeds. If any of multiple `$N` clauses matches, the rules will fail.

11.6.11 Destination-Channel-Specific Rewrite Rules (`$C`, `$Q`)

It is possible to have rewrite rules whose application is dependent upon the channel to which a message is being enqueued. This is useful when there are two names for some host, one known to one group of hosts and one known to another. By using different channels to send mail to each group, addresses can be rewritten to refer to the host under the name known to each group.

Destination channel-specific rewriting is associated with the channel to which the message is to be dequeued and processed by, and the channel keywords `rules` and `norules` on that channel. If `norules` is specified on the destination channel, no channel-specific rewrite checking is done. If `rules` is specified on the destination channel, channel-specific rule checks are enforced. The keyword `rules` is the default.

Destination channel-specific rewriting is not associated with the channel matched by a given address. It depends only on the message's envelope `To:` address. When a message is enqueued, its envelope `To:` address is first rewritten to determine to which channel the message is enqueued. During the rewriting of the envelope `To:` address, any `$C` and `$Q` control sequences are ignored. After the envelope `To:` address is rewritten and the destination channel determined, then the `$C` and `$Q` control sequences are honored, as other addresses associated with the message are rewritten.

Destination-channel-specific rewrite checking is triggered by the presence of a `$C` or `$Q` control sequence in the template part of a rule. The characters following the `$C` or `$Q`, up until either an at sign (@), percent sign (%), or subsequent `$N`, `$M`, `$C`, `$Q`, `$T`, or `$?` are interpreted as a channel name.

For example, `$Qchannel` causes the rule to fail if *channel* is not the destination. For another example, `$Cchannel` causes the rule to fail if *channel* is the destination. Multiple `$Q` and `$C` clauses may be specified. If any one of multiple `$Q` clauses matches, the rule succeeds. If any of multiple `$C` clauses matches, the rule fails.

11.6.12 Direction-and-Location-Specific Rewrite Rules (`$B`, `$E`, `$F`, `$R`)

Sometimes you need to specify rewrite rules that apply only to envelope addresses or, alternately, only to header addresses. The control sequence `$E` forces a rewrite to fail if the address being rewritten is not an envelope address. The control sequence `$B` forces a rewrite to fail if the address being rewritten is not from the message header or body. These sequences have no other effects on the rewrite and may appear anywhere in the rewrite rule template.

Addresses may also be categorized by direction. A forward pointing address is one that originates on a `To:`, `Cc:`, `Resent-to:`, or other header or envelope line that refers to a destination. A backward pointing address is something like a `From:`, `Sender:`, or `Resent-From:`, that refers to a source. The control sequence `$F` causes the rewrite to be applied if the address is forward pointing. The control sequence `$R` causes the rewrite to be applied if the address is reverse pointing.

11.6.13 Host-Location-Specific Rewrites (\$A, \$P, \$S, \$X)

Circumstances occasionally require rewriting that is sensitive to the location where a host name appears in an address. Host names can appear in several different contexts in an address:

- In a source route
- To the right of the at sign (@)
- To the right of a percent sign (%) in the local-part
- To the left of an exclamation point in the local-part

Under normal circumstances, a host name should be handled in the same way, regardless of where it appears. Some situations might require specialized handling.

Four control sequences are used to control matching on the basis of the host's location in the address.

- \$S specifies that the rule can match a host extracted from a source route.
- \$A specifies that the rule can match a host found to the right of the @ sign.
- \$P specifies that the rule can match a host found to the right of a % sign.
- \$X specifies that the rule can match a host found to the left of an exclamation point (!).

The rule fails if the host is from a location other than the one specified. These sequences can be combined in a single rewrite rule. For example, if \$S and \$A are specified, the rule matches hosts specified in either a source route or to the right of the at sign. Specifying none of these sequences is equivalent to specifying all of them; the rule can match regardless of location.

11.6.14 Changing the Current Tag Value, \$T

The \$T control sequence is used to change the current rewrite rule tag. The rewrite rule tag is prepended to all rewrite rule patterns before they are looked up in the configuration file and domain database. Text following the \$T, up until either an at sign, percent sign, \$N, \$M, \$Q, \$C, \$T, or \$? is taken to be the new tag.

Tags are useful in handling special addressing forms where the entire nature of an address is changed when a certain component is encountered. For example, suppose that the special host name internet, when found in a source route, should be removed from the address and the resulting address forcibly matched against the TCP-DAEMON channel.

This could be implemented with rules like the following (localhost is assumed to be the official name of the local host):

```
internet      $$@localhost$Tmtcp-force|
```

```
mtcp-force|.    $U%$H@TCP-DAEMON
```

The first rule will match the special host name internet if it appears in the source route. It forcibly matches internet against the local channel, which insures that it will be removed from the address. A rewrite tag is then set. Rewriting proceeds, but no regular rule will match because of the tag. Finally, the default rule is tried with the tag, and the second rule of this set fires, forcibly matching the address against the TCP-DAEMON channel regardless of any other criteria.

11.6.15 Controlling Error Messages Associated with Rewriting (\$?)

The MTA provides default error messages when rewriting and channel matching fail. The ability to change these messages can be useful under certain circumstances. For example, if someone tries to send mail to an Ethernet router box, it may be considered more informative to say something like “our routers cannot accept mail” rather than the usual “illegal host/domain specified.”

A special control sequence can be used to change the error message that is printed if the rule fails. The sequence \$? is used to specify an error message. Text following the \$?, up to either an at sign (@), percent sign (%), \$N, \$M, \$Q, \$C, \$T, or \$? is taken to be the text of the error message to print if the result of this rewrite fails to match any channel. The setting of an error message is “sticky” and lasts through the rewriting process.

A rule that contains a \$? operates just like any other rule. The special case of a rule containing only a \$? and nothing else receives special attention --- the rewriting process is terminated without changing the mailbox or host portions of the address and the host is looked up as-is in the channel table. This lookup is expected to fail and the error message will be returned as a result.

For example, assume the final rewrite rule in the MTA configuration file is as follows:

```
. $?Unrecognized address; contact postmaster@siroe.com
```

In this example, any unrecognized host or domain specifications that can fail will, in the process of failing, generate the error message: Unrecognized address; contact postmaster@siroe.com.

11.7 Handling Large Numbers of Rewrite Rules

The MTA always reads in all the rewrite rules from the `imta.cnf` file and stores them in memory in a hash table. Use of a compiled configuration bypasses the overhead associated with reading the configuration file each and every time the information is needed; a hash table is still used to store all of the rewrite rules in memory. This scheme is adequate for small to medium numbers of rewrite rules. However, some sites may require as many as 10,000 rewrite rules or more, which can consume prohibitive amounts of memory.

The MTA solves this problem by providing an optional facility for storing large numbers of rewrite rules in an ancillary indexed data file. Whenever the regular configuration file is read, the MTA checks for the existence of the domain database. If this database exists, it is opened and consulted whenever an attempted match fails on the rules found in the configuration file. The domain database is only checked if a given rule is not found in the configuration file, so rules can always be added to the configuration file to override those in the database. By default, the domain database is used to store rewrite rules associated with hosted domains. The `IMTA_DOMAIN_DATABASE` attribute is stored in the `imta_tailor` file. The default location for the database is `msg-svr-base/data/db/domaindb.db`.

Note – DO NOT EDIT THIS FILE BY HAND.

11.8 Testing Rewrite Rules

You can test rewrite rules with the `imsimta test -rewrite` command. The `-noimage` qualifier will allow you to test changes made to the configuration file prior to recompiling the new configuration.

You may find it helpful to rewrite a few addresses using this utility with the `-debug` qualifier. This will show you step-by-step how the address is rewritten. For example, issue the following command:

```
% imsimta test -rewrite -debug joe@siroe.com
```

For a detailed description of the `imsimta test -rewrite` utility, see the *Sun Java System Messaging Server 6.3 Administration Reference*.

11.9 Rewrite Rules Example

The following example provides sample rewrite rules and how sample addresses would be rewritten by the rules.

Suppose the configuration file for the system SC.CS.SIROE.EDU contained the rewrite rules shown in the following example:

```

sc           $U@sc.cs.siroe.edu
sc1          $U@sc1.cs.siroe.edu
sc2          $U@sc2.cs.siroe.edu
*            $U%$&0.cs.siroe.edu
*.cs         $U%$&0.cs.siroe.edu
*.cs.siroe   $U%$&0.cs.siroe.edu
*.cs.siroe.edu $U%$&0.cs.siroe.edu@ds.adm.siroe.edu
sc.cs.siroe.edu $U@$D
sc1.cs.siroe.edu $U@$D
sc2.cs.siroe.edu $U@$D
sd.cs.siroe.edu $U@sd.cs.siroe.edu
.siroe.edu    $U%$H.siroe.edu@cds.adm.siroe.edu
.edu         $U@$H$D@gate.adm.siroe.edu
[]           $U@[ $L ]@gate.adm.siroe.edu

```

Table 11–7 shows some sample addresses and how they would be rewritten and routed according to the rewrite rules.

TABLE 11–7 Sample Addresses and Rewrites

Initial address	Rewritten as	Routed to
user@sc	user@sc.cs.siroe.edu	sc.cs.siroe.edu
user@sc1	user@sc1.cs.siroe.edu	sc1.cs.siroe.edu
user@sc2	user@sc2.cs.siroe.edu	sc2.cs.siroe.edu
user@sc.cs	user@sc.cs.siroe.edu	sc.cs.siroe.edu
user@sc1.cs	user@sc1.cs.siroe.edu	sc1.cs.siroe.edu
user@sc2.cs	user@sc2.cs.siroe.edu	sc2.cs.siroe.edu
user@sc.cs.siroe	user@sc.cs.siroe.edu	sc.cs.siroe.edu
user@sc1.cs.siroe	user@sc1.cs.siroe.edu	sc1.cs.siroe.edu
user@sc2.cs.siroe	user@sc2.cs.siroe.edu	sc2.cs.siroe.edu
user@sc.cs.siroe.edu	user@sc.cs.siroe.edu	sc.cs.siroe.edu
user@sc1.cs.siroe.edu	user@sc1.cs.siroe.edu	sc1.cs.siroe.edu

TABLE 11-7 Sample Addresses and Rewrites *(Continued)*

Initial address	Rewritten as	Routed to
user@sc2.cs.siroe.edu	user@sc2.cs.siroe.edu	sc2.cs.siroe.edu
user@sd.cs.siroe.edu	user@sd.cs.siroe.edu	sd.cs.siroe.edu
user@aa.cs.siroe.edu	user@aa.cs.siroe.edu	ds.adm.siroe.edu
user@a.eng.siroe.edu	user@a.eng.siroe.edu	cds.adm.siroe.edu
user@a.cs.sesta.edu	user@a.cs.sesta.edu	gate.adm.siroe.edu—route inserted
user@b.cs.sesta.edu	user@b.cs.sesta.edu	gate.adm.siroe.edu— route inserted
user@[1.2.3.4]	user@[1.2.3.4]	gate.adm.siroe.edu—route inserted

Basically, what these rewrite rules say is: If the host name is one of our short-form names (sc, sc1 or sc2) or if it is one of our full names (sc.cs.siroe.edu, and so on), expand it to our full name and route it to us. Append cs.cmu.edu to one part short-form names and try again. Convert one part followed by .cs to one part followed by .cs.siroe.edu and try again. Also convert .cs.siroe to .cs.siroe.edu and try again.

If the name is sd.cs.siroe.edu (some system we connect to directly, perhaps) rewrite and route it there. If the host name is anything else in the .cs.siroe.edu subdomain, route it to ds.cs.siroe.edu (the gateway for the .cs.siroe.edu subdomain). If the host name is anything else in the .siroe.edu subdomain route it to cds.adm.siroe.edu (the gateway for the .siroe.edu subdomain). If the host name is anything else in the .edu top-level domain route it to gate.adm.siroe.edu (which is presumably capable of routing the message to its proper destination). If a domain literal is used send it to gate.adm.siroe.edu as well.

Most applications of rewrite rules (like the previous example) will not change the username (or mailbox) part of the address in any way. The ability to change the username part of the address is used when the MTA is used to interface to mailers that do not conform to RFC 822—mailers where it is necessary to stuff portions of the host/domain specification into the username part of the address. This capability should be used with great care if it is used at all.

Configuring Channel Definitions

This chapter describes how to use channel keyword definitions in the MTA configuration file `imta.cnf`. Please read [Chapter 10, “About MTA Services and Configuration,”](#) as well as [“8.5.3 Channel Definitions” on page 200](#) and [“10.2 The MTA Configuration File” on page 235](#) before reading this chapter. This chapter contains the following sections:

- [“12.1 Configuring Channel Defaults” on page 318](#)
- [“12.2 Channel Keywords Listed Alphabetically” on page 318](#)
- [“12.3 Channel Keywords Categorized by Function” on page 330](#)
- [“12.4 Configuring SMTP Channels” on page 359](#)
- [“12.5 Configuring Message Processing and Delivery” on page 380](#)
- [“12.6 Configuring Address Handling” on page 389](#)
- [“12.7 Configuring Header Handling” on page 400](#)
- [“12.8 Attachments and MIME Processing” on page 405](#)
- [“12.9 Limits on Messages, Quotas, Recipients, and Authentication Attempts” on page 410](#)
- [“12.10 File Creation in the MTA Queue” on page 415](#)
- [“12.11 Configuring Logging and Debugging” on page 417](#)
- [“12.12 Miscellaneous Keywords” on page 418](#)

Note – If you make channel definition changes in `imta.cnf`, you must restart any programs or channels that load the configuration data only once when they start up—for example, the SMTP server—by using the `imsimta restart` command. If you are using a compiled configuration, you must recompile and then restart. For more information about compiling configuration information and starting programs, see the [Sun Java System Messaging Server 6.3 Administration Reference](#).

12.1 Configuring Channel Defaults

Many configurations involve repetition of various channel keywords on all or nearly all channels. Maintaining such a configuration is both tedious and error-prone. To simplify some configurations, you can specify which keywords are defaults for various channels.

For example, the following line in a configuration file indicates that all channel blocks following the line will inherit the keywords specified in the line:

```
defaults keyword1 keyword2 keyword3 ...
```

The `defaults` line can be thought of as a special channel block that changes the keyword defaults without actually specifying a channel. The `defaults` line also does not require any additional lines of channel block information (if any are specified they will be ignored).

There is no limit on the number of `defaults` lines that can be specified—the effects of multiple `defaults` lines are cumulative with the most recently encountered (reading from top to bottom) line having precedence.

It may be useful to unconditionally eliminate the effects of any `defaults` lines starting at some point in the configuration file (at the start of a standalone section of channel blocks in an external file, for example). The `nодеfaults` line is provided for this purpose. For example, inserting the following line in the configuration file nullifies all settings established by any previous `defaults` channel and returns the configuration to the state that would apply if no `defaults` had been specified:

```
nодеfaults
```

Like regular channel blocks, a blank line must separate each `defaults` or `nодеfaults` channel block from other channel blocks. The `defaults` and `nodefаults` channel blocks are the only channel blocks which may appear before the local channel in the configuration file. However, like any other channel block, they must appear after the last rewrite rule.

12.2 Channel Keywords Listed Alphabetically

The following table is an alphabetized list of keywords.

TABLE 12-1 Alphabetized List of Channel Keywords

Keyword	For More Information...
733	“12.6.1 Address Types and Conventions” on page 389
822	“12.6.1 Address Types and Conventions” on page 389

TABLE 12-1 Alphabetized List of Channel Keywords *(Continued)*

Keyword	For More Information...
addreturnpath	“12.6.11 Generating of Return-path Header Lines” on page 396
addressssrs	“15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518
addrspersfile	“12.9.4 Handling Mail Delivery to Over Quota Users” on page 414
Aliasdetourhost	“12.12.6 Routing After Address Validation But Before Expansion” on page 421
aliaslocal	“12.6.15 Specifying Alias File and Alias Database Probes” on page 398
aliaspostmaster	“Postmaster Returned Message Content” on page 281
allowetrn	“12.4.2.3 ETRN Command Support” on page 364
allowswitchchannel	“12.4.3.8 Alternate Channels for Incoming Mail (Switch Channels)” on page 374
alternatechannel	“12.9.3 Retargeting Messages Exceeding Limit on Size or Recipients” on page 412
alternateblocklimit	“12.9.3 Retargeting Messages Exceeding Limit on Size or Recipients” on page 412
alternatelinelimit	“12.9.3 Retargeting Messages Exceeding Limit on Size or Recipients” on page 412
alternaterecipientlimit	“12.9.3 Retargeting Messages Exceeding Limit on Size or Recipients” on page 412
authrewrite	“12.4.3 TCP/IP Connection and DNS Lookup Support” on page 368
backoff	“12.5.3 Specifying the Retry Frequency for Messages that Failed Delivery” on page 383
bangoverpercent	“12.6.3 Adding Routing Information in Addresses” on page 391
bangstyle	“12.6.1 Address Types and Conventions” on page 389
bidirectional	“12.5.1 Setting Channel Directionality” on page 382
blocketrn	“12.4.2.3 ETRN Command Support” on page 364
blocklimit	“12.9.2 Specifying Absolute Message Size Limits” on page 411
cacheeverything	“12.4.3.2 Caching for Channel Connection Information” on page 371
cachefailures	“12.4.3.2 Caching for Channel Connection Information” on page 371
cachesuccesses	“12.4.3.2 Caching for Channel Connection Information” on page 371
caption	“12.12.9 Set Channel Displays for Monitoring Framework” on page 425
channelfilter	“12.12.4 Specifying Mailbox Filter File Location” on page 419
charset7	“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366
charset8	“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366

TABLE 12-1 Alphabetized List of Channel Keywords *(Continued)*

Keyword	For More Information...
charsetesc	“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366
checkehlo	“12.4.2.2 EHLO Command Support” on page 363
chunkingclient	“12.4.6 Support for SMTP Chunking” on page 378
chunkingserver	“12.4.6 Support for SMTP Chunking” on page 378
commentinc	“12.6.13 Handling Comments in Address Header Lines” on page 396
commentmap	“12.6.13 Handling Comments in Address Header Lines” on page 396
commentomit	“12.6.13 Handling Comments in Address Header Lines” on page 396
commentstrip	“12.6.13 Handling Comments in Address Header Lines” on page 396
commenttotal	“12.6.13 Handling Comments in Address Header Lines” on page 396
connectalias	“12.6.5 Address Rewriting Upon Message Dequeue” on page 393
connectcanonical	“12.6.5 Address Rewriting Upon Message Dequeue” on page 393
copysendpost	“Returned Failed Messages” on page 280
copywarnpost	“Warning Messages” on page 280
daemon	“12.4.3.10 Target Host Choice” on page 375
datefour	“12.7.4 Converting Date to Two- or Four-Digits” on page 402
datetwo	“12.7.4 Converting Date to Two- or Four-Digits” on page 402
dayofweek	“12.7.5 Specifying Day of Week in Date” on page 403
defaulthost	“12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses” on page 393
defaultmx	“12.4.3.5 TCP/IP MX Record Support” on page 373
defaultnameservers	“12.4.3.6 Nameserver Lookups” on page 373
deferralrejectlimit	“12.12.8 Setting Limits on Bad RCPT TO Addresses” on page 425
deferred	“12.5.2 Implementing Deferred Delivery Dates” on page 382
defragment	“12.8.2 Automatic Defragmentation of Message/Partial Messages” on page 406
dequeue_removeroute	“12.6.18 Removing Source Routes” on page 399
description	“12.12.9 Set Channel Displays for Monitoring Framework” on page 425
destinationfilter	“12.12.4 Specifying Mailbox Filter File Location” on page 419
destinationnosolicit	“12.12.7 NO-SOLICIT SMTP Extension Support” on page 424

TABLE 12-1 Alphabetized List of Channel Keywords *(Continued)*

Keyword	For More Information...
destinationspamfilterX	“12.12.5 Spam Filter Keywords” on page 420
destinationspamfilterXoptin	“12.12.5 Spam Filter Keywords” on page 420
destinationsrs	“15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518
disabledestinationspamfilterX	“12.12.5 Spam Filter Keywords” on page 420
disableetrn	“12.4.2.3 ETRN Command Support” on page 364
disablesourcespamfilterX	“12.12.5 Spam Filter Keywords” on page 420
dispositionchannel	“12.12.1 Process Channel Overrides” on page 418
disconnectbadauthlimit	“12.9.1 Limits on Unsuccessful Authentication Attempts” on page 410
disconnectbadcommandlimit	“12.10.3 Setting Session Limits” on page 416
domainetrn	“12.4.2.3 ETRN Command Support” on page 364
domainvrfy	“12.4.2.4 VRFY Command Support” on page 365
dropblank	“12.6.8 Stripping Illegal Blank Recipient Headers” on page 395
ehlo	“12.4.2.2 EHLO Command Support” on page 363
eightbit	“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366
eightnegotiate	“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366
eightstrict	“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366
errsendpost	“Returned Failed Messages” on page 280
errwarnpost	“Warning Messages” on page 280
expandchannel	“12.5.9 Expansion of Multiple Addresses” on page 388
expandlimit	“12.5.9 Expansion of Multiple Addresses” on page 388
expnallow	“12.4.2.5 EXPN Support” on page 365
expndisable	“12.4.2.5 EXPN Support” on page 365
expndefault	“12.4.2.5 EXPN Support” on page 365
exproute	“12.6.3 Adding Routing Information in Addresses” on page 391
fileinto	“12.12.4 Specifying Mailbox Filter File Location” on page 419
filesperjob	“12.5.5 Service Job Limits” on page 384
filter	“12.12.4 Specifying Mailbox Filter File Location” on page 419

TABLE 12-1 Alphabetized List of Channel Keywords *(Continued)*

Keyword	For More Information...
forwardcheckdelete	“12.4.3.3 Reverse DNS Lookups” on page 371
forwardchecknone	“12.4.3.3 Reverse DNS Lookups” on page 371
forwardchecktag	“12.4.3.3 Reverse DNS Lookups” on page 371
header_733	“12.6.1 Address Types and Conventions” on page 389
header_822	“12.6.1 Address Types and Conventions” on page 389
header_uucp	“12.6.1 Address Types and Conventions” on page 389
headerlabelalign	“12.7.7 Header Alignment and Folding” on page 403
headerlimit	“12.9.8 Limiting Header Size” on page 415
headerlinelength	“12.7.7 Header Alignment and Folding” on page 403
headerread	“12.7.2 Removing Selected Message Header Lines” on page 401
headertrim	“12.7.2 Removing Selected Message Header Lines” on page 401
holdexquota	“12.9.4 Handling Mail Delivery to Over Quota Users” on page 414
holdlimit	“12.5.9 Expansion of Multiple Addresses” on page 388
identnone	“12.4.3.4 IDENT Lookups” on page 372
identnonelimited	“12.4.3.4 IDENT Lookups” on page 372
identnonenumeric	“12.4.3.4 IDENT Lookups” on page 372
identnonesymbolic	“12.4.3.4 IDENT Lookups” on page 372
identtcp	“12.4.3.4 IDENT Lookups” on page 372
identtcplimited	“12.4.3.4 IDENT Lookups” on page 372
identtcpsymbolic	“12.4.3.4 IDENT Lookups” on page 372
ignoreencoding	“12.8.1 Ignoring the Encoding Header Line” on page 406
ignoremessageencoding	“12.8.5 Interpreting Content-transfer-encoding Fields on Multiparts and Message/RFC822 Parts” on page 410
ignoremultipartencoding	“12.8.5 Interpreting Content-transfer-encoding Fields on Multiparts and Message/RFC822 Parts” on page 410
improute	“12.6.3 Adding Routing Information in Addresses” on page 391
includefinal	“10.10.4.4 To Include Altered Addresses in Status Notification Messages” on page 279
inenttcpnumeric	“12.4.3.4 IDENT Lookups” on page 372

TABLE 12-1 Alphabetized List of Channel Keywords *(Continued)*

Keyword	For More Information...
inner	“12.7.1 Rewriting Embedded Headers” on page 401
innertrim	“12.7.2 Removing Selected Message Header Lines” on page 401
interfaceaddress	“12.4.3.1 TCP/IP Port Number and Interface Address” on page 370
interpretencoding	“12.8.1 Ignoring the Encoding Header Line” on page 406
interpretmessageencoding	“12.8.5 Interpreting Content-transfer-encoding Fields on Multiparts and Message/RFC822 Parts” on page 410
interpretmultipartencoding	“12.8.5 Interpreting Content-transfer-encoding Fields on Multiparts and Message/RFC822 Parts” on page 410
language	“12.7.10 Setting Default Language in Headers” on page 405
lastresort	“12.4.3.7 Last Resort Host” on page 374
linelength	“12.8.4 Imposing Message Line Length Restrictions” on page 409
linelimit	“12.9.2 Specifying Absolute Message Size Limits” on page 411
localvrfy	“12.4.2.4 VRFY Command Support” on page 365
logging	“12.11.1 Logging Keywords” on page 417
logheader	“12.11.1 Logging Keywords” on page 417
loopcheck	“12.11.3 Setting Loopcheck” on page 418
mailfromdnsverify	“12.4.2.6 DNS Domain Verification” on page 366
master	“12.5.1 Setting Channel Directionality” on page 382
master_debug	“12.11.2 Debugging Keywords” on page 417
maxblocks	“12.8.3 Automatic Fragmentation of Large Messages” on page 408
maxheaderaddrs	“12.7.6 Automatic Splitting of Long Header Lines” on page 403
maxheaderchars	“12.7.6 Automatic Splitting of Long Header Lines” on page 403
maxjobs	“12.5.5 Service Job Limits” on page 384
maxlines	“12.8.3 Automatic Fragmentation of Large Messages” on page 408
maxprocchars	“12.7.7 Header Alignment and Folding” on page 403
maysaslserver	“12.4.4 SMTP Authentication, SASL, and TLS” on page 376
maytls	“12.4.8 Transport Layer Security” on page 379
maytlsclient	“12.4.8 Transport Layer Security” on page 379

TABLE 12-1 Alphabetized List of Channel Keywords *(Continued)*

Keyword	For More Information...
maytlsserver	“12.4.8 Transport Layer Security” on page 379
missingrecipientpolicy	“12.6.7 Legalizing Messages Without Recipient Header Lines” on page 394
msexchange	“12.4.7 Specifying Microsoft Exchange Gateway Channels” on page 379
multiple	“12.4.3.10 Target Host Choice” on page 375
mustsaslserver	“12.4.4 SMTP Authentication, SASL, and TLS” on page 376
musttls	“12.4.8 Transport Layer Security” on page 379
musttlsclient	“12.4.8 Transport Layer Security” on page 379
musttlsserver	“12.4.8 Transport Layer Security” on page 379
mx	“12.4.3.5 TCP/IP MX Record Support” on page 373
namelengthlimit	“12.9.6 Controlling the Length of General and Filename Content-type and Content-disposition Parameters” on page 414
nameservers	“12.4.3.6 Nameserver Lookups” on page 373
noaddresssrs	“15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518
noaddrreturnpath	“12.6.11 Generating of Return-path Header Lines” on page 396
nobangoverpercent	“12.6.3 Adding Routing Information in Addresses” on page 391
noblocklimit	“12.9.2 Specifying Absolute Message Size Limits” on page 411
nocache	“12.4.3.2 Caching for Channel Connection Information” on page 371
nochannelfilter	“12.12.4 Specifying Mailbox Filter File Location” on page 419
nochunkingclient	“12.4.6 Support for SMTP Chunking” on page 378
nochunkingserver	“12.4.6 Support for SMTP Chunking” on page 378
nodayofweek	“12.7.5 Specifying Day of Week in Date” on page 403
nodefaulthost	“12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses” on page 393
nodeferred	“12.5.2 Implementing Deferred Delivery Dates” on page 382
nodefragment	“12.8.2 Automatic Defragmentation of Message/Partial Messages” on page 406
nodeestinationfilter	“12.12.4 Specifying Mailbox Filter File Location” on page 419
nodeestinationssrs	“15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518

TABLE 12-1 Alphabetized List of Channel Keywords *(Continued)*

Keyword	For More Information...
nodropblank	“12.6.8 Stripping Illegal Blank Recipient Headers” on page 395
noehlo	“12.4.2.2 EHLO Command Support” on page 363
noexproute	“12.6.3 Adding Routing Information in Addresses” on page 391
noexquota	“12.9.4 Handling Mail Delivery to Over Quota Users” on page 414
nofileinto	“12.12.4 Specifying Mailbox Filter File Location” on page 419
nofilter	“12.12.4 Specifying Mailbox Filter File Location” on page 419
noheaderread	“12.7.2 Removing Selected Message Header Lines” on page 401
noheadertrim	“12.7.2 Removing Selected Message Header Lines” on page 401
noimproute	“12.6.3 Adding Routing Information in Addresses” on page 391
noinner	“12.7.1 Rewriting Embedded Headers” on page 401
noinnertrim	“12.7.2 Removing Selected Message Header Lines” on page 401
nolinelimit	“12.9.2 Specifying Absolute Message Size Limits” on page 411
nologging	“12.11.1 Logging Keywords” on page 417
noloopcheck	“12.11.3 Setting Loopcheck” on page 418
nomailfromdnsverify	“12.4.2.6 DNS Domain Verification” on page 366
nomaster_debug	“12.11.2 Debugging Keywords” on page 417
nomsexchange	“12.4.3 TCP/IP Connection and DNS Lookup Support” on page 368
nomx	“12.4.3.5 TCP/IP MX Record Support” on page 373
norandomemx	“12.4.3.5 TCP/IP MX Record Support” on page 373
nosourcesrs	“15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518
nonurgentbackoff	“12.5.3 Specifying the Retry Frequency for Messages that Failed Delivery” on page 383
nonurgentblocklimit	“12.5.7 Message Priority Based on Size” on page 387
nonurgentnotices	“10.10.4.3 To Set Notification Message Delivery Intervals” on page 278
noreceivedfor	“12.6.12 Constructing Received Header Lines from Envelope To and From Addresses” on page 396
noreceivedfrom	“12.6.12 Constructing Received Header Lines from Envelope To and From Addresses” on page 396

TABLE 12-1 Alphabetized List of Channel Keywords *(Continued)*

Keyword	For More Information...
noremotehost	“12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses” on page 393
norestricted	“12.6.10 Enabling Restricted Mailbox Encoding” on page 395
noreturnaddress	“Postmaster Returned Message Content” on page 281
noreturnpersonal	“Postmaster Returned Message Content” on page 281
noreverse	“12.6.9 Enabling Channel-Specific Use of the Reverse Database” on page 395
normalbackoff	“12.5.3 Specifying the Retry Frequency for Messages that Failed Delivery” on page 383
normalblocklimit	“12.5.7 Message Priority Based on Size” on page 387
normalnotices	“10.10.4.3 To Set Notification Message Delivery Intervals” on page 278
norules	“12.6.17 Enabling Channel-specific Rewrite Rules Checks” on page 399
nosasl	“12.4.4 SMTP Authentication, SASL, and TLS” on page 376
nosaslserver	“12.4.4 SMTP Authentication, SASL, and TLS” on page 376
nosaslswitchchannel	“12.4.4 SMTP Authentication, SASL, and TLS” on page 376
nosendetrn	“12.4.2.3 ETRN Command Support” on page 364
nosendpost	“Returned Failed Messages” on page 280
noservice	“12.5.10 Enable Service Conversions” on page 389
noslave_debug	“12.11.2 Debugging Keywords” on page 417
nosmtplib	“12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362
nosourcefilter	“12.12.4 Specifying Mailbox Filter File Location” on page 419
noswitchchannel	“12.4.3.8 Alternate Channels for Incoming Mail (Switch Channels)” on page 374
notices	“10.10.4.3 To Set Notification Message Delivery Intervals” on page 278
notificationchannel	“12.12.1 Process Channel Overrides” on page 418
notls	“12.4.8 Transport Layer Security” on page 379
notlsclient	“12.4.8 Transport Layer Security” on page 379
notlsserver	“12.4.8 Transport Layer Security” on page 379
novrfy	“12.4.2.4 VRFY Command Support” on page 365
nowarnpost	“Warning Messages” on page 280

TABLE 12-1 Alphabetized List of Channel Keywords *(Continued)*

Keyword	For More Information...
nox_env_to	“12.7.3 Generating/Removing X-Envelope-to Header Lines” on page 402
parameterlengthlimit	“12.9.6 Controlling the Length of General and Filename Content-type and Content-disposition Parameters” on page 414
percentonly	“12.6.3 Adding Routing Information in Addresses” on page 391
percents	“12.6.1 Address Types and Conventions” on page 389
personalinc	“12.6.14 Handling Personal Names in Address Header Lines” on page 397
personalmap	“12.6.14 Handling Personal Names in Address Header Lines” on page 397
personalomit	“12.6.14 Handling Personal Names in Address Header Lines” on page 397
personalstrip	“12.6.14 Handling Personal Names in Address Header Lines” on page 397
pool	“12.5.4 Processing Pools for Channel Execution Jobs” on page 384
port	“12.4.3.1 TCP/IP Port Number and Interface Address” on page 370
postheadbody	“Postmaster Returned Message Content” on page 281
postheadonly	“Postmaster Returned Message Content” on page 281
randommx	“12.4.3.5 TCP/IP MX Record Support” on page 373
receivedfor	“12.6.12 Constructing Received Header Lines from Envelope To and From Addresses” on page 396
receivedfrom	“12.6.12 Constructing Received Header Lines from Envelope To and From Addresses” on page 396
recipientcutoff	“12.9.7 Limiting Message Recipients” on page 414
recipientlimit	“12.9.7 Limiting Message Recipients” on page 414
rejectsmtplonglines	“12.9.5 Handling SMTP Mail with Lines Exceeding 1000 Characters” on page 414
remotehost	“12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses” on page 393
restricted	“12.6.10 Enabling Restricted Mailbox Encoding” on page 395
returnaddress	“Postmaster Returned Message Content” on page 281
returnenvelope	“Blank Envelope Return Addresses” on page 280
returnpersonal	“Postmaster Returned Message Content” on page 281
reverse	“12.6.9 Enabling Channel-Specific Use of the Reverse Database” on page 395
routelocal	“12.6.4 Disabling Rewriting of Explicit Routing Addresses” on page 392

TABLE 12-1 Alphabetized List of Channel Keywords *(Continued)*

Keyword	For More Information...
rules	“12.6.17 Enabling Channel-specific Rewrite Rules Checks” on page 399
saslswitchchannel	“12.4.4 SMTP Authentication, SASL, and TLS” on page 376
sendetrn	“12.4.2.3 ETRN Command Support” on page 364
sendpost	“Returned Failed Messages” on page 280
sensitivitycompanyconfidential	“12.7.9 Sensitivity Checking” on page 404
sensitivitynormal	“12.7.9 Sensitivity Checking” on page 404
sensitivitypersonal	“12.7.9 Sensitivity Checking” on page 404
sensitivityprivate	“12.7.9 Sensitivity Checking” on page 404
service	“12.5.10 Enable Service Conversions” on page 389
sevenbit	“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366
silentetrn	“12.4.2.3 ETRN Command Support” on page 364
single	“12.4.3.10 Target Host Choice” on page 375
single_sys	“12.4.3.10 Target Host Choice” on page 375
slave	“12.5.1 Setting Channel Directionality” on page 382
slave_debug	“12.11.2 Debugging Keywords” on page 417
smtp	“12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362
smtp_cr	“12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362
smtp_crlf	“12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362
smtp_crnlf	“12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362
smtp_lf	“12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362
sourceblocklimit	“12.9.2 Specifying Absolute Message Size Limits” on page 411
sourcecommentinc	“12.6.13 Handling Comments in Address Header Lines” on page 396
sourcecommentmap	“12.6.13 Handling Comments in Address Header Lines” on page 396
sourcecommenttomit	“12.6.13 Handling Comments in Address Header Lines” on page 396
sourcecommentstrip	“12.6.13 Handling Comments in Address Header Lines” on page 396
sourcecommenttotal	“12.6.13 Handling Comments in Address Header Lines” on page 396
sourcefilter	“12.12.4 Specifying Mailbox Filter File Location” on page 419
sourcenosolicit	“12.12.7 NO-SOLICIT SMTP Extension Support” on page 424

TABLE 12-1 Alphabetized List of Channel Keywords *(Continued)*

Keyword	For More Information...
sourcepersonalinc	“12.6.14 Handling Personal Names in Address Header Lines” on page 397
sourcepersonalmap	“12.6.14 Handling Personal Names in Address Header Lines” on page 397
sourcepersonalomit	“12.6.14 Handling Personal Names in Address Header Lines” on page 397
sourcepersonalstrip	“12.6.14 Handling Personal Names in Address Header Lines” on page 397
sourceroute	“12.6.1 Address Types and Conventions” on page 389
sourcespamfilterX	“12.12.5 Spam Filter Keywords” on page 420
sourcespamfilterXoptin	“12.12.5 Spam Filter Keywords” on page 420
sourcesrs	“15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518
streaming	“12.4.2.8 Protocol Streaming” on page 368
subaddressexact	“12.6.16 Subaddress Handling” on page 398
subaddressrelaxed	“12.6.16 Subaddress Handling” on page 398
subaddresswild	“12.6.16 Subaddress Handling” on page 398
subdirs	“12.10.2 Spreading a Channel Message Queue Across Multiple Subdirectories” on page 416
submit	“12.12.2 Channel Operation Type” on page 419
suppressfinal	“10.10.4.4 To Include Altered Addresses in Status Notification Messages” on page 279
switchchannel	“12.4.3.8 Alternate Channels for Incoming Mail (Switch Channels)” on page 374
threaddepth	“12.5.8 SMTP Channel Threads” on page 387
tlsswitchchannel	“12.4.8 Transport Layer Security” on page 379
transactionlimit	“12.5.6 Setting Connection Transaction Limits” on page 386
truncatesmtplonglines	“12.9.5 Handling SMTP Mail with Lines Exceeding 1000 Characters” on page 414
unrestricted	“12.6.10 Enabling Restricted Mailbox Encoding” on page 395
urgentbackoff	“12.5.3 Specifying the Retry Frequency for Messages that Failed Delivery” on page 383
urgentblocklimit	“12.5.7 Message Priority Based on Size” on page 387
urgentnotices	“10.10.4.3 To Set Notification Message Delivery Intervals” on page 278
useintermediate	“10.10.4.4 To Include Altered Addresses in Status Notification Messages” on page 279

TABLE 12-1 Alphabetized List of Channel Keywords *(Continued)*

Keyword	For More Information...
user	“12.12.3 Pipe Channel” on page 419
userswitchchannel	“12.4.3.9 Source Channel Switching Based on User or Domain Settings” on page 375
uucp	“12.6.1 Address Types and Conventions” on page 389
viaaliasoptional	“12.6.19 Specifying Address Must be from an Alias” on page 400
viaaliasrequired	“12.6.19 Specifying Address Must be from an Alias” on page 400
vrfyallow	“12.4.2.4 VRFY Command Support” on page 365
vrfydefault	“12.4.2.4 VRFY Command Support” on page 365
vrfyhide	“12.4.2.4 VRFY Command Support” on page 365
warnpost	“Warning Messages” on page 280
wrapsmtplonglines	“12.9.5 Handling SMTP Mail with Lines Exceeding 1000 Characters” on page 414
x_env_to	“12.7.3 Generating/Removing X-Envelope-to Header Lines” on page 402

12.3 Channel Keywords Categorized by Function

The following tables are categorized lists of keywords. The tables and categories are as follows:

- [Table 12-2 Address Handling Keywords](#)
- [Table 12-3 Attachments and MIME Processing](#)
- [Table 12-4 Character Sets and Eight Bit Data](#)
- [Table 12-5 File Creation in the MTA Queue Area](#)
- [Table 12-6 Header Keywords](#)
- [Table 12-7 Incoming Channel Matching and Switching Keywords](#)
- [Table 12-8 Logging and Debugging Channel Keywords](#)
- [Table 12-9 Long Address Lists or Headers Channel Keywords](#)
- [Table 12-10 Mailbox Filter Channel Keywords](#)
- [Table 12-11 NO-SOLICIT SMTP Extension Support Keywords](#)
- [Table 12-12 Notification and Postmaster Messages Keywords](#)
- [Table 12-13 Processing Control and Job Submission Keywords](#)
- [Table 12-14 Sensitivity Limit Keywords](#)
- [Table 12-15 Keywords for Limits on Messages, User Quotas, Privileges, and Authentication Attempts](#)

- [Table 12–16](#) SMTP Authentication, SASL and TLS Keywords
- [Table 12–17](#) SMTP Commands and Protocol Keywords
- [Table 12–18](#) TCP/IP Connection and DNS Lookup Support Keywords
- [Table 12–19](#) Miscellaneous Keywords

TABLE 12–2 Address Handling Keywords

Keyword	Definition
Address Handling	
733	Use % routing in the envelope; synonymous with percents. “ 12.6.1 Address Types and Conventions ” on page 389
822	“ 12.6.1 Address Types and Conventions ” on page 389 Use source routes in the envelope; same as <code>sourceroute</code> .
addreturnpath	“ 12.6.11 Generating of Return-path Header Lines ” on page 396 Add <code>Return-path:</code> header to messages enqueued to this channel.
aliaslocal	“ 12.6.15 Specifying Alias File and Alias Database Probes ” on page 398 Look up rewritten addresses in the alias file and alias database.
authrewrite	“ 12.4.3 TCP/IP Connection and DNS Lookup Support ” on page 368 Used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers.
bangoverpercent	“ 12.6.3 Adding Routing Information in Addresses ” on page 391 Group A!B%C as A!(B%C)
bangstyle	“ 12.6.1 Address Types and Conventions ” on page 389 Use UUCP ! routing in the envelope; synonymous with <code>uucp</code> .
defaulthost	“ 12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses ” on page 393 Specify a domain name to use to complete addresses
dequeue_removeroute	“ 12.6.18 Removing Source Routes ” on page 399 Removes source routes from envelope <code>To:</code> addresses.
exproute	“ 12.6.3 Adding Routing Information in Addresses ” on page 391 Require explicit routing when addresses passed to remote systems.
holdlimit	“ 12.5.9 Expansion of Multiple Addresses ” on page 388 Hold message when number of envelope recipient addresses exceeds this limit.

TABLE 12-2 Address Handling Keywords *(Continued)*

Keyword	Definition
improute	<p>“12.6.3 Adding Routing Information in Addresses” on page 391</p> <p>Implicit routing for this channel’s addresses</p>
missingrecipientpolicy	<p>“12.6.7 Legalizing Messages Without Recipient Header Lines” on page 394</p> <p>Set policy for how to legalize (which header to add) messages that are lacking any recipient headers.</p>
noaddrreturnpath	<p>“12.6.11 Generating of Return-path Header Lines” on page 396</p> <p>Do not add Return-path: header when enqueueing message.</p>
nobangoverpercent	<p>“12.6.3 Adding Routing Information in Addresses” on page 391</p> <p>Group A!B%C as (A!B)%C</p>
nodefaulthost	<p>“12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses” on page 393</p> <p>Do not specify a domain name to use to complete addresses</p>
noexproute	<p>“12.6.3 Adding Routing Information in Addresses” on page 391</p> <p>No explicit routing for this channel’s addresses</p>
noimproute	<p>“12.6.3 Adding Routing Information in Addresses” on page 391</p> <p>No implicit routing for this channel’s addresses</p>
noreceivedfrom	<p>“12.6.12 Constructing Received Header Lines from Envelope To and From Addresses” on page 396</p> <p>Construct Received: header lines without including the original envelope From: address.</p>
norremotehost	<p>“12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses” on page 393</p> <p>Use local host’s domain name as the default domain name to complete addresses</p>
norestricted	<p>“12.6.10 Enabling Restricted Mailbox Encoding” on page 395</p> <p>Same as <code>unrestricted</code>.</p>
noreverse	<p>“12.6.9 Enabling Channel-Specific Use of the Reverse Database” on page 395</p> <p>Exempts addresses in messages from address reversal processing</p>
norules	<p>“12.6.17 Enabling Channel-specific Rewrite Rules Checks” on page 399</p> <p>Do not enforce channel-specific rewrite rule checks for this channel.</p>
percentonly	<p>“12.6.3 Adding Routing Information in Addresses” on page 391</p> <p>Ignores bang paths. Use % routing in the envelope.</p>

TABLE 12-2 Address Handling Keywords *(Continued)*

Keyword	Definition
percents	<p>“12.6.1 Address Types and Conventions” on page 389</p> <p>Use % routing in the envelope; synonymous with 733.</p>
remotehost	<p>“12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses” on page 393</p> <p>Use remote host’s name as the default domain name to complete addresses</p>
restricted	<p>“12.6.10 Enabling Restricted Mailbox Encoding” on page 395</p> <p>The channel connects to mail systems that require encoding.</p>
reverse	<p>“12.6.9 Enabling Channel-Specific Use of the Reverse Database” on page 395</p> <p>Checked addresses against address reversal database or REVERSE mapping</p>
routelocal	<p>“12.6.4 Disabling Rewriting of Explicit Routing Addresses” on page 392</p> <p>Causes the MTA, when rewriting an address to the channel, to attempt to “short circuit” any explicit routing in the address.</p>
rules	<p>“12.6.17 Enabling Channel-specific Rewrite Rules Checks” on page 399</p> <p>Enforce channel-specific rewrite rule checks for this channel.</p>
sourceroute	<p>“12.6.1 Address Types and Conventions” on page 389</p> <p>Synonymous with 822.</p>
subaddressexact	<p>“12.6.16 Subaddress Handling” on page 398</p> <p>Perform no special subaddress handling during entry matching; the entire mailbox, including the subaddress, must match an entry in order for the alias to be considered to match.</p>
subaddressrelaxed	<p>“12.6.16 Subaddress Handling” on page 398</p> <p>After looking for an exact match and then a match of the form name+*, the MTA should make one additional check for a match on just the name portion.</p>
subaddresswild	<p>“12.6.16 Subaddress Handling” on page 398</p> <p>After looking for an exact match including the entire subaddress, the MTA should next look for an entry of the form name+*.</p>
unrestricted	<p>“12.6.10 Enabling Restricted Mailbox Encoding” on page 395</p> <p>Tells the MTA not to perform RFC 1137 encoding and decoding.</p>
uucp	<p>“12.6.1 Address Types and Conventions” on page 389</p> <p>Use UUCP! routing in the envelope; synonymous with bangstyle.</p>

TABLE 12-2 Address Handling Keywords *(Continued)*

Keyword	Definition
viaaliasoptional	<p>“12.6.19 Specifying Address Must be from an Alias” on page 400</p> <p>Final recipient addresses that match the channel are not required to be produced by an alias.</p>
viaaliasrequired	<p>“12.6.19 Specifying Address Must be from an Alias” on page 400</p> <p>Final recipient address that matches the channel must be produced by an alias.</p>

TABLE 12-3 Attachments and MIME Processing

Keyword	Definition
defragment	<p>“12.8.2 Automatic Defragmentation of Message/Partial Messages” on page 406</p> <p>Partial messages queued to the channel are placed in the defragmentation channel queue instead.</p>
ignoreencoding	<p>“12.8.1 Ignoring the Encoding Header Line” on page 406</p> <p>Ignore Encoding: header on incoming messages.</p>
ignoremessageencoding	<p>“12.8.5 Interpreting Content-transfer-encoding Fields on Multiparts and Message/RFC822 Parts” on page 410</p> <p>Ignore content-transfer-encoding fields on message/rfc822 parts of incoming messages.</p>
ignoremultipartencoding	<p>“12.8.5 Interpreting Content-transfer-encoding Fields on Multiparts and Message/RFC822 Parts” on page 410</p> <p>Ignore content-transfer-encoding fields on multiparts parts of incoming messages.</p>
interpretencoding	<p>“12.8.1 Ignoring the Encoding Header Line” on page 406</p> <p>Interpret Encoding: header on incoming messages, if the need arises.</p>
interpretmessageencoding	<p>“12.8.5 Interpreting Content-transfer-encoding Fields on Multiparts and Message/RFC822 Parts” on page 410</p> <p>Interpret content-transfer-encoding fields on message/rfc822 parts of incoming messages.</p>
interpretmultipartencoding	<p>“12.8.5 Interpreting Content-transfer-encoding Fields on Multiparts and Message/RFC822 Parts” on page 410</p> <p>Interpret content-transfer-encoding fields on multiparts parts of incoming messages.</p>
nodefragment	<p>“12.8.2 Automatic Defragmentation of Message/Partial Messages” on page 406</p> <p>Disables defragmentation.</p>

TABLE 12-4 Character Sets and Eighth Bit Data

Keyword	Definition
charset7	<p>“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366</p> <p>Default character set to associate with 7-bit text messages</p>
charset8	<p>“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366</p> <p>Default character set to associate with 8-bit text messages</p>
charsetsc	<p>“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366</p> <p>Default character set to associate with 7-bit text containing the escape character</p>
eightbit	<p>“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366</p> <p>Channel supports eight-bit characters.</p>
eightnegotiate	<p>“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366</p> <p>Channel should negotiate use of eight-bit transmission if possible.</p>
eightstrict	<p>“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366</p> <p>Reject messages with headers that contain unnegotiated eight-bit data.</p>
sevenbit	<p>“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366</p> <p>Do not support 8-bit characters; 8-bit characters must be encoded.</p>

TABLE 12-5 File Creation in the MTA Queue Area

Keyword	Page	Definition
addrsprofile	“12.9.4 Handling Mail Delivery to Over Quota Users” on page 414	Limit on the maximum number of recipients that can be associated with a single message file in a channel queue
expandchannel	“12.5.9 Expansion of Multiple Addresses” on page 388	Specifies channel in which to perform deferred expansion due to application of expandlimit.
expandlimit	“12.5.9 Expansion of Multiple Addresses” on page 388	Processes an incoming message “off-line” when the number of addressees exceeds this limit.
multiple	“12.9.4 Handling Mail Delivery to Over Quota Users” on page 414	No limit on the number of recipients in a message file, however the SMTP channel defaults to 99.
single	“12.9.4 Handling Mail Delivery to Over Quota Users” on page 414	A separate copy of the message will be created for each destination address on the channel.

TABLE 12–5 File Creation in the MTA Queue Area *(Continued)*

Keyword	Page	Definition
single_sys	“12.9.4 Handling Mail Delivery to Over Quota Users” on page 414	Create a single message copy for each destination system used.
subdirs	“12.10.2 Spreading a Channel Message Queue Across Multiple Subdirectories” on page 416	Specifies the number of subdirectories across which to spread messages for the channel queues.

TABLE 12–6 Header Keywords

Keyword	Definition
authrewrite	“12.4.3 TCP/IP Connection and DNS Lookup Support” on page 368 Used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers.
commentinc	“12.6.13 Handling Comments in Address Header Lines” on page 396 Leave comments in message header lines intact.
commentmap	“12.6.13 Handling Comments in Address Header Lines” on page 396 Runs comment strings in message header lines through the <code>COMMENT_STRINGS</code> mapping table.
commentomit	“12.6.13 Handling Comments in Address Header Lines” on page 396 Remove comments from message header lines.
commentstrip	“12.6.13 Handling Comments in Address Header Lines” on page 396 Remove problematic characters from comment fields in message header lines.
commenttotal	“12.6.13 Handling Comments in Address Header Lines” on page 396 Strip comments (material in parentheses) from all header lines, except Received: header lines. Not recommended.
datefour	“12.7.4 Converting Date to Two- or Four-Digits” on page 402 Expand all year fields to four digits.
datetwo	“12.7.4 Converting Date to Two- or Four-Digits” on page 402 Remove the leading two digits from four-digit dates. Provides compatibility with mail systems that require two digit dates; it should never be used for any other purpose.
dayofweek	“12.7.5 Specifying Day of Week in Date” on page 403 Retain day of the week information and adds this information to date and time headers if it is missing.
defaulthost	“12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses” on page 393 Specify a domain name to use to complete addresses

TABLE 12-6 Header Keywords *(Continued)*

Keyword	Definition
deletemessagehash	<p>“12.7.11 Controlling Message-hash: Headers” on page 405</p> <p>Delete any existing Message-hash: field.</p>
dropblank	<p>“12.6.8 Stripping Illegal Blank Recipient Headers” on page 395</p> <p>Strip illegal blank headers from incoming messages.</p>
generatemessagehash	<p>“12.7.11 Controlling Message-hash: Headers” on page 405</p> <p>If specified on a destination channel, it causes a Message-hash: header field to be inserted into the message.</p>
header_733	<p>“12.6.1 Address Types and Conventions” on page 389</p> <p>Use % routing in the message header.</p>
header_822	<p>“12.6.1 Address Types and Conventions” on page 389</p> <p>Use source routes in the message header.</p>
headerlabelalign	<p>“12.7.7 Header Alignment and Folding” on page 403</p> <p>Controls the alignment point for message headers enqueued on this channel; it takes an integer-valued argument.</p>
headerlinelength	<p>“12.7.7 Header Alignment and Folding” on page 403</p> <p>Controls the length of header lines enqueued on this channel.</p>
headerread	<p>“12.7.2 Removing Selected Message Header Lines” on page 401</p> <p>Apply header trimming rules from an options file to the message headers upon message enqueue (use with caution) before the original message headers are processed.</p>
headertrim	<p>“12.7.2 Removing Selected Message Header Lines” on page 401</p> <p>Applies header trimming rules from an options file to the message headers after the original message headers are processed.</p>
header_uucp	<p>“12.6.1 Address Types and Conventions” on page 389</p> <p>Use ! routing in the header</p>
inner	<p>“12.7.1 Rewriting Embedded Headers” on page 401</p> <p>Parse messages and rewrite inner headers.</p>
innertrim	<p>“12.7.2 Removing Selected Message Header Lines” on page 401</p> <p>Apply header trimming rules from an options file to inner message headers (use with caution).</p>
keepmessagehash	<p>“12.7.11 Controlling Message-hash: Headers” on page 405</p> <p>Causes any existing Message-hash: field to be retained.</p>

TABLE 12-6 Header Keywords (*Continued*)

Keyword	Definition
language	<p>“12.7.10 Setting Default Language in Headers” on page 405</p> <p>Specifies the default language in headers.</p>
maxheaderaddrs	<p>“12.7.6 Automatic Splitting of Long Header Lines” on page 403</p> <p>Controls how many addresses can appear on a single line.</p>
maxheaderchars	<p>“12.7.6 Automatic Splitting of Long Header Lines” on page 403</p> <p>Controls how many characters can appear on a single line.</p>
missingrecipientpolicy	<p>“12.6.7 Legalizing Messages Without Recipient Header Lines” on page 394</p> <p>Set policy for how to legalize (which header to add) messages that are lacking any recipient headers.</p>
nodayofweek	<p>“12.7.5 Specifying Day of Week in Date” on page 403</p> <p>Removes day of the week from date and time headers. Provides compatibility with mail systems that cannot process this information; it should never be used for any other purpose.</p>
nodefaulthost	<p>“12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses” on page 393</p> <p>Do not specify a domain name to use to complete addresses</p>
nodropblank	<p>“12.6.8 Stripping Illegal Blank Recipient Headers” on page 395</p> <p>Do not strip illegal blank headers from incoming messages.</p>
noheaderread	<p>“12.7.2 Removing Selected Message Header Lines” on page 401</p> <p>Do not apply header trimming rules from option file.</p>
noheadertrim	<p>“12.7.2 Removing Selected Message Header Lines” on page 401</p> <p>Do not apply header trimming rules from options file.</p>
noinner	<p>“12.7.1 Rewriting Embedded Headers” on page 401</p> <p>Do not rewrite inner message header lines.</p>
noinnertrim	<p>“12.7.2 Removing Selected Message Header Lines” on page 401</p> <p>Do not apply header trimming to inner message headers.</p>
noreceivedfor	<p>“12.6.12 Constructing Received Header Lines from Envelope To and From Addresses” on page 396</p> <p>Construct Received: header lines without including any envelope recipient information.</p>
noreceivedfrom	<p>“12.6.12 Constructing Received Header Lines from Envelope To and From Addresses” on page 396</p> <p>Construct Received: header lines without including the original envelope From: address.</p>
noremotehost	<p>“12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses” on page 393</p> <p>Use local host’s domain name as the default domain name to complete addresses</p>

TABLE 12-6 Header Keywords *(Continued)*

Keyword	Definition
noreverse	<p>“12.6.9 Enabling Channel-Specific Use of the Reverse Database” on page 395</p> <p>Exempts addresses in messages queued to the channel from address reversal processing</p>
norules	<p>“12.6.17 Enabling Channel-specific Rewrite Rules Checks” on page 399</p> <p>Do not enforce channel-specific rewrite rule checks for this channel.</p>
nox_env_to	<p>“12.7.3 Generating/Removing X-Envelope-to Header Lines” on page 402</p> <p>Remove X-Envelope-to header lines.</p>
personalinc	<p>“12.6.14 Handling Personal Names in Address Header Lines” on page 397</p> <p>Leave personal name fields in message header lines intact.</p>
personalmap	<p>“12.6.14 Handling Personal Names in Address Header Lines” on page 397</p> <p>Run personal names through PERSONAL_NAMES mapping table.</p>
personalomit	<p>“12.6.14 Handling Personal Names in Address Header Lines” on page 397</p> <p>Remove personal name fields from message header lines.</p>
personalstrip	<p>“12.6.14 Handling Personal Names in Address Header Lines” on page 397</p> <p>Strip problem characters from personal name fields in header lines.</p>
receivedfor	<p>“12.6.12 Constructing Received Header Lines from Envelope To and From Addresses” on page 396</p> <p>If a message is addressed to just one envelope recipient, to include that envelope To: address in the Received: header line it constructs.</p>
receivedfrom	<p>“12.6.12 Constructing Received Header Lines from Envelope To and From Addresses” on page 396</p> <p>Include the original envelope From: address when constructing a Received: header line for an incoming message if the MTA has changed the envelope From: address.</p>
remotehost	<p>“12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses” on page 393</p> <p>Use remote host's name as the default domain name to complete addresses</p>
restricted	<p>“12.6.10 Enabling Restricted Mailbox Encoding” on page 395</p> <p>Channel connects to mail systems that require this encoding.</p>
reverse	<p>“12.6.9 Enabling Channel-Specific Use of the Reverse Database” on page 395</p> <p>Check addresses against address reversal database or REVERSE mapping</p>
rules	<p>“12.6.17 Enabling Channel-specific Rewrite Rules Checks” on page 399</p> <p>Enforce channel-specific rewrite rule checks for this channel.</p>

TABLE 12-6 Header Keywords (*Continued*)

Keyword	Definition
sensitivitycompanyconfidential	“12.7.9 Sensitivity Checking” on page 404 Companyconfidential is the upper sensitivity limit of messages accepted.
sensitivitynormal	“12.7.9 Sensitivity Checking” on page 404 Normal is the upper sensitivity limit of messages accepted.
sensitivitypersonal	“12.7.9 Sensitivity Checking” on page 404 Personal is the upper sensitivity limit of messages accepted.
sensitivityprivate	“12.7.9 Sensitivity Checking” on page 404 Private is the upper sensitivity limit of messages accepted.
sourcecommentinc	“12.6.13 Handling Comments in Address Header Lines” on page 396 Leave comments in incoming message header lines.
sourcecommentmap	“12.6.13 Handling Comments in Address Header Lines” on page 396 Runs comment strings in header lines through source channels.
sourcecommentomit	“12.6.13 Handling Comments in Address Header Lines” on page 396 Remove comments from incoming message header lines, for example, To:, From:, and Cc: headers.
sourcecommentstrip	“12.6.13 Handling Comments in Address Header Lines” on page 396 Remove problematic characters from comment field in incoming header lines.
sourcecommenttotal	“12.6.13 Handling Comments in Address Header Lines” on page 396 Strip comments (material in parentheses) in incoming messages.
sourcepersonalinc	“12.6.14 Handling Personal Names in Address Header Lines” on page 397 Leave personal names in incoming message header lines intact.
sourcepersonalmap	“12.6.14 Handling Personal Names in Address Header Lines” on page 397 Run personal names through source channels.
sourcepersonalomit	“12.6.14 Handling Personal Names in Address Header Lines” on page 397 Remove personal name fields from incoming message header lines.
sourcepersonalstrip	“12.6.14 Handling Personal Names in Address Header Lines” on page 397 Strip problematic characters from personal name fields in incoming message header lines.
unrestricted	“12.6.10 Enabling Restricted Mailbox Encoding” on page 395 Tells the MTA not to perform RFC 1137 encoding and decoding.

TABLE 12-6 Header Keywords *(Continued)*

Keyword	Definition
x_env_to	<p>“12.7.3 Generating/Removing X-Envelope-to Header Lines” on page 402</p> <p>Enables generation of X-Envelope-to header lines.</p>

TABLE 12-7 Incoming Channel Matching and Switching Keywords

Keyword	Definition
allowswitchchannel	<p>“12.4.3.8 Alternate Channels for Incoming Mail (Switch Channels)” on page 374</p> <p>Allows switching to this channel from a switchchannel channel</p>
nosaslswitchchannel	<p>“12.4.4 SMTP Authentication, SASL, and TLS” on page 376</p> <p>No switching to this channel upon successful SASL authentication</p>
noswitchchannel	<p>“12.4.3.8 Alternate Channels for Incoming Mail (Switch Channels)” on page 374</p> <p>No channel switching should be done to or from the channel.</p>
switchchannel	<p>“12.4.3.8 Alternate Channels for Incoming Mail (Switch Channels)” on page 374</p> <p>Switches from the server channel to the channel associated with the originating host.</p>
saslswitchchannel	<p>“12.4.4 SMTP Authentication, SASL, and TLS” on page 376</p> <p>Cause incoming connections to be switched to a specified channel upon a client’s successful use of SASL.</p>
tlsswitchchannel	<p>“12.4.8 Transport Layer Security” on page 379</p> <p>Switches to another channel upon successful TLS negotiation.</p>
userswitchchannel	<p>“12.4.3.9 Source Channel Switching Based on User or Domain Settings” on page 375</p> <p>Switches source channel based on user or domain settings.</p>

TABLE 12-8 Logging and Debugging Channel Keywords

Keyword	Definition
logging	<p>“12.11.1 Logging Keywords” on page 417</p> <p>Log message enqueues and dequeues into the log file and activates logging for a particular channel.</p>
loopcheck	<p>“12.11.3 Setting Loopcheck” on page 418</p> <p>Places a string into the SMTP EHLO response banner in order for the MTA to check if it is communicating with itself.</p>
master_debug	<p>“12.11.2 Debugging Keywords” on page 417</p> <p>Create debugging output in the channel’s master program output.</p>

TABLE 12–8 Logging and Debugging Channel Keywords *(Continued)*

Keyword	Definition
nologging	“12.11.1 Logging Keywords” on page 417 Do not log message enqueues and dequeues into the log file.
noloopcheck	“12.11.3 Setting Loopcheck” on page 418 No string into the SMTP EHLO response banner.
nomaster_debug	“12.11.2 Debugging Keywords” on page 417 No debugging output in the channel’s master program output.
noslave_debug	“12.11.2 Debugging Keywords” on page 417 Do not generate slave debugging output.
slave_debug	“12.11.2 Debugging Keywords” on page 417 Generate slave debug output.

TABLE 12–9 Long Address Lists or Headers Channel Keywords

Keyword	Definition
expandchannel	“12.5.9 Expansion of Multiple Addresses” on page 388 Specifies channel in which to perform deferred expansion due to application of expandlimit.
expandlimit	“12.5.9 Expansion of Multiple Addresses” on page 388 Processes an incoming message “off-line” when the number of addressees exceeds this limit.
holdlimit	“12.5.9 Expansion of Multiple Addresses” on page 388 Holds a message when the number of addresses exceeds this limit.
maxprocchars	“12.7.7 Header Alignment and Folding” on page 403 Maximum length header that can be processed and rewritten.

TABLE 12–10 Mailbox Filter Channel Keywords

Keyword	Definition
channelfilter	“12.12.4 Specifying Mailbox Filter File Location” on page 419 Location of channel filter file; same as destinationfilter.
destinationfilter	Location of channel filter file that applies to outgoing messages.
destinationspamfilterX	“12.12.5 Spam Filter Keywords” on page 420 Run messages destined for this channel through spam filtering software X. Does not accept spam filtering software parameters.

TABLE 12-10 Mailbox Filter Channel Keywords *(Continued)*

Keyword	Definition
destinationspamfilterXoptin	“12.12.5 Spam Filter Keywords” on page 420 Run messages destined to this channel through spam filtering software X.
disabledestinationspamfilterX	“12.12.5 Spam Filter Keywords” on page 420 Disables spam filter X for messages destined to this channel.
disablesourcespamfilterX	“12.12.5 Spam Filter Keywords” on page 420 Disables spam filter X for messages coming from this channel.
fileinto	“12.12.4 Specifying Mailbox Filter File Location” on page 419 Specify effect on address when a mailbox filter fileinto operation is applied.
filter	“12.12.4 Specifying Mailbox Filter File Location” on page 419 Specify the location of user filter files.
nochannelfilter	“12.12.4 Specifying Mailbox Filter File Location” on page 419 No channel filtering for outgoing messages. Also known as nodestinationfilter.
nodestinationfilter	“12.12.4 Specifying Mailbox Filter File Location” on page 419 Do not perform channel filtering for outgoing messages.
nofileinto	“12.12.4 Specifying Mailbox Filter File Location” on page 419 Mailbox filter fileinto operator has no effect.
nofilter	“12.12.4 Specifying Mailbox Filter File Location” on page 419 Do not perform user mailbox filtering.
nosourcefilter	“12.12.4 Specifying Mailbox Filter File Location” on page 419 Do not perform channel filtering for incoming messages.
sourcefilter	“12.12.4 Specifying Mailbox Filter File Location” on page 419 Specify the location of channel filter file for incoming messages.
sourcespamfilterX	“12.12.5 Spam Filter Keywords” on page 420 Run messages originating from this channel through spam filtering software X. Does not accept spam filtering software parameters.
sourcespamfilterXoptin	“12.12.5 Spam Filter Keywords” on page 420 Run messages originating from this channel through spam filtering software X. Accepts spam filtering software parameters.

TABLE 12-11 NO-SOLICIT SMTP Extension Support Keywords

Keyword	Definition
sourcenosolicit	<p>“12.12.7 NO-SOLICIT SMTP Extension Support” on page 424</p> <p>Specifies a comma-separated list of solicitation field values that will be blocked in mail submitted by this channel.</p>
destinationnosolicit	<p>“12.12.7 NO-SOLICIT SMTP Extension Support” on page 424</p> <p>Specifies a comma-separated list of solicitation field values that will not be accepted in mail queued to this channel.</p>

TABLE 12-12 Notification and Postmaster Messages Keywords

Keyword	Definition
(See page “10.10 Controlling Delivery Status Notification Messages” on page 271 for complete notification procedures)	
aliaspostmaster	<p>“Postmaster Returned Message Content” on page 281</p> <p>Messages addressed to the user name postmaster at the official channel name are redirected to postmaster@local-host, where local-host is the local host name (the name on the local channel).</p>
copysendpost	<p>“Returned Failed Messages” on page 280</p> <p>Sends a copy of the failure notice to the postmaster unless the originator address on the failing message is blank.</p>
copywarnpost	<p>“Warning Messages” on page 280</p> <p>Sends a copy of the warning message to the postmaster unless the originator address on the undelivered message is blank.</p>
errsendpost	<p>“Returned Failed Messages” on page 280</p> <p>Sends a copy of the failure notice to the postmaster only when the notice cannot be returned to the originator.</p>
errwarnpost	<p>“Warning Messages” on page 280</p> <p>Sends a copy of the warning message to the postmaster when the notice cannot be returned to the originator.</p>
includefinal	<p>“10.10.4.4 To Include Altered Addresses in Status Notification Messages” on page 279</p> <p>Include final form of recipient address in delivery notifications.</p>
nonurgentnotices	<p>“10.10.4.3 To Set Notification Message Delivery Intervals” on page 278</p> <p>Specifies the amount of time that may elapse before notices are sent and messages returned for messages of non-urgent priority.</p>
noreturnaddress	<p>“Postmaster Returned Message Content” on page 281</p> <p>Use RETURN_ADDRESS option value as postmaster address name.</p>

TABLE 12-12 Notification and Postmaster Messages Keywords *(Continued)*

Keyword	Definition
noreturnpersonal	<p>“Postmaster Returned Message Content” on page 281</p> <p>Use RETURN_PERSONAL option value as postmaster personal name.</p>
normalnotices	<p>“10.10.4.3 To Set Notification Message Delivery Intervals” on page 278</p> <p>Specifies the amount of time that may elapse before notices are sent and messages returned for messages of normal priority.</p>
nosendpost	<p>“Returned Failed Messages” on page 280</p> <p>Disables sending a copy of all failed messages to the postmaster.</p>
notices	<p>“10.10.4.3 To Set Notification Message Delivery Intervals” on page 278</p> <p>Specifies the amount of time that may elapse before notices are sent and messages returned.</p>
nowarnpost	<p>“Warning Messages” on page 280</p> <p>Disables sending a copy of warning messages to the postmaster.</p>
postheadbody	<p>“Postmaster Returned Message Content” on page 281</p> <p>Returns both the headers and the contents of the message.</p>
postheadonly	<p>“Postmaster Returned Message Content” on page 281</p> <p>Returns only headers to the postmaster.</p>
returnaddress	<p>“Postmaster Returned Message Content” on page 281</p> <p>Specifies the return address for the local postmaster.</p>
returnenvelope	<p>“Blank Envelope Return Addresses” on page 280</p> <p>Control use of blank envelope return addresses.</p>
returnpersonal	<p>“Postmaster Returned Message Content” on page 281</p> <p>Set the personal name for the local postmaster.</p>
sendpost	<p>“Returned Failed Messages” on page 280</p> <p>Enables sending a copy of all failed messages to the postmaster.</p>
suppressfinal	<p>“10.10.4.4 To Include Altered Addresses in Status Notification Messages” on page 279</p> <p>Suppress the final address form from notification messages, if an original address form is present, from notification messages.</p>
urgentnotices	<p>“10.10.4.3 To Set Notification Message Delivery Intervals” on page 278</p> <p>Specify the amount of time which may elapse before notices are sent and messages returned for messages of urgent priority.</p>

TABLE 12–12 Notification and Postmaster Messages Keywords *(Continued)*

Keyword	Definition
useintermediate	<p>“10.10.4.4 To Include Altered Addresses in Status Notification Messages” on page 279</p> <p>Uses an intermediate form of the address produced after list expansion, but prior to user mailbox name generation.</p>
warnpost	<p>“Warning Messages” on page 280</p> <p>Enables sending a copy of warning messages to the postmaster.</p>

TABLE 12–13 Processing Control and Job Submission Keywords

Keyword	Definition
(See “ 12.5 Configuring Message Processing and Delivery ” on page 380 for greater functional granularity)	
backoff	<p>“12.5.3 Specifying the Retry Frequency for Messages that Failed Delivery” on page 383</p> <p>Frequency of attempted redelivery of unsuccessfully delivered messages. Can be overridden by the keywords <code>normalbackoff</code>, <code>nonurgentbackoff</code>, <code>urgentbackoff</code>.</p>
bidirectional	<p>“12.5.1 Setting Channel Directionality” on page 382</p> <p>Channel served by a master and slave program.</p>
deferred	<p>“12.5.2 Implementing Deferred Delivery Dates” on page 382</p> <p>Recognize and honor of the <code>Deferred-delivery</code>: header line.</p>
expandchannel	<p>“12.5.9 Expansion of Multiple Addresses” on page 388</p> <p>Specifies channel in which to perform deferred expansion due to application of <code>expandlimit</code>.</p>
expandlimit	<p>“12.5.9 Expansion of Multiple Addresses” on page 388</p> <p>Processes an incoming message “off-line” when the number of addressees exceeds this limit.</p>
filesperjob	<p>“12.5.5 Service Job Limits” on page 384</p> <p>Number of queue entries to be processed by a single job.</p>
master	<p>“12.5.1 Setting Channel Directionality” on page 382</p> <p>Channel served by a master program (<code>master</code>).</p>
maxjobs	<p>“12.5.5 Service Job Limits” on page 384</p> <p>Maximum number of jobs that can be running concurrently for the channel.</p>
nodeferred	<p>“12.5.2 Implementing Deferred Delivery Dates” on page 382</p> <p>Specifies that <code>Deferred-delivery</code>: header line not be honored.</p>
nonurgentbackoff	<p>“12.5.3 Specifying the Retry Frequency for Messages that Failed Delivery” on page 383</p> <p>The frequency for attempted redelivery of nonurgent messages.</p>

TABLE 12-13 Processing Control and Job Submission Keywords *(Continued)*

Keyword	Definition
nonurgentblocklimit	<p>“12.5.7 Message Priority Based on Size” on page 387</p> <p>Forces messages above this size to lower than nonurgent priority (second class priority), meaning that the messages will always wait for the next periodic job for further processing.</p>
normalbackoff	<p>“12.5.3 Specifying the Retry Frequency for Messages that Failed Delivery” on page 383</p> <p>The frequency for attempted redelivery of normal messages.</p>
normalblocklimit	<p>“12.5.7 Message Priority Based on Size” on page 387</p> <p>Forces messages above this size to nonurgent priority.</p>
noservice	<p>“12.5.10 Enable Service Conversions” on page 389</p> <p>Service conversions for messages coming into this channel must be enabled via CHARSET - CONVERSION.</p>
pool	<p>“12.5.4 Processing Pools for Channel Execution Jobs” on page 384</p> <p>Specifies a pool for a channel. Must be followed by the pool name to which delivery jobs for the current channel should be pooled.</p>
service	<p>“12.5.10 Enable Service Conversions” on page 389</p> <p>Unconditionally enables service conversions regardless of CHARSET - CONVERSION entry.</p>
slave	<p>“12.5.1 Setting Channel Directionality” on page 382</p> <p>Channel served by a slave program (slave).</p>
threaddepth	<p>“12.5.8 SMTP Channel Threads” on page 387</p> <p>Number of messages triggering new thread with multithreaded SMTP client.</p>
transactionlimit	Limits the number of messages allowed per connection.
urgentbackoff	<p>“12.5.3 Specifying the Retry Frequency for Messages that Failed Delivery” on page 383</p> <p>The frequency for attempted redelivery of urgent messages.</p>
urgentblocklimit	<p>“12.5.7 Message Priority Based on Size” on page 387</p> <p>Forces messages above this size to normal priority.</p>
user	<p>“12.12.3 Pipe Channel” on page 419</p> <p>Used on pipe channels to indicate under what user name to run.</p>

TABLE 12-14 Sensitivity Limit Keywords

Keyword	Definition
sensitivitycompanyconfidential	<p>“12.7.9 Sensitivity Checking” on page 404</p> <p>Upper sensitivity limit of messages accepted.</p>

TABLE 12-14 Sensitivity Limit Keywords *(Continued)*

Keyword	Definition
sensitivitynormal	<p>“12.7.9 Sensitivity Checking” on page 404</p> <p>Normal is the upper sensitivity limit of messages accepted.</p>
sensitivitypersonal	<p>“12.7.9 Sensitivity Checking” on page 404</p> <p>Personal is the upper sensitivity limit of messages accepted.</p>
sensitivityprivate	<p>“12.7.9 Sensitivity Checking” on page 404</p> <p>Private is the upper sensitivity limit of messages accepted.</p>

TABLE 12-15 Keywords for Limits on Messages, User Quotas, Privileges, and Authentication Attempts

Keyword	Definition
alternatechannel	<p>“12.9.3 Retargeting Messages Exceeding Limit on Size or Recipients” on page 412</p> <p>Alternate destination channel for alternateblocklimit, alternatelinelimit, and alternaterecipientlimit</p>
alternateblocklimit	<p>“12.9.3 Retargeting Messages Exceeding Limit on Size or Recipients” on page 412</p> <p>Specifies limit on the number of blocks in a message before it will be sent to alternativechannel.</p>
alternatelinelimit	<p>“12.9.3 Retargeting Messages Exceeding Limit on Size or Recipients” on page 412</p> <p>Specifies limit on the number of lines in a message before it will be sent to alternativechannel.</p>
alternaterecipientlimit	<p>“12.9.3 Retargeting Messages Exceeding Limit on Size or Recipients” on page 412</p> <p>Specifies limit on the number of recipients in a message before it will be sent to alternativechannel.</p>
blocklimit	<p>“12.9.2 Specifying Absolute Message Size Limits” on page 411</p> <p>Maximum number of MTA blocks allowed per message.</p>
disconnectbadauthlimit	<p>“12.9.1 Limits on Unsuccessful Authentication Attempts” on page 410</p> <p>Limit on the number of unsuccessful authentication attempts that will be allowed in a session before the session is disconnected.</p>
disconnectbadcommandlimit	<p>“12.10.3 Setting Session Limits” on page 416</p> <p>Limits the number of session bad commands.</p>
disconnectrecipientlimit	<p>“12.10.3 Setting Session Limits” on page 416</p> <p>Limits the number of session recipients.</p>
disconnectrejectlimit	<p>“12.10.3 Setting Session Limits” on page 416</p> <p>Limits the number of rejected recipients.</p>

TABLE 12-15 Keywords for Limits on Messages, User Quotas, Privileges, and Authentication Attempts *(Continued)*

Keyword	Definition
disconnecttransactionlimit	“12.10.3 Setting Session Limits” on page 416 Limits the number of transactions.
headerlimit	“12.9.8 Limiting Header Size” on page 415 Limit on the maximum size of the primary (outermost) message header
holdexquota	“12.9.4 Handling Mail Delivery to Over Quota Users” on page 414 Hold messages for users that are over quota.
holdlimit	“12.5.9 Expansion of Multiple Addresses” on page 388 Holds an incoming message when the number of addresses exceeds this limit.
linelength	“12.8.4 Imposing Message Line Length Restrictions” on page 409 Limits the maximum permissible message line length on a channel-by-channel basis.
linelimit	“12.9.2 Specifying Absolute Message Size Limits” on page 411 Maximum number of lines allowed per message.
maxblocks	“12.8.3 Automatic Fragmentation of Large Messages” on page 408 Specifies the maximum number of blocks allowed in a message.
maxlines	“12.8.3 Automatic Fragmentation of Large Messages” on page 408 Specifies the maximum number of lines allowed in a message.
nameparameterlengthlimit	“12.9.6 Controlling the Length of General and Filename Content-type and Content-disposition Parameters” on page 414 Controls the points at which the name content-type and filename content-disposition parameters are truncated.
noblocklimit	“12.9.2 Specifying Absolute Message Size Limits” on page 411 No limit for the number of MTA blocks allowed per message.
noexquota	“12.9.4 Handling Mail Delivery to Over Quota Users” on page 414 Return to originator any messages to users who are over quota.
nolinelimit	“12.9.2 Specifying Absolute Message Size Limits” on page 411 No limit specified for the number of lines allowed per message.
nonurgentblocklimit	“12.5.7 Message Priority Based on Size” on page 387 Forces messages above this size to lower than nonurgent priority (second class priority), meaning that the messages will always wait for the next periodic job for further processing.

TABLE 12-15 Keywords for Limits on Messages, User Quotas, Privileges, and Authentication Attempts *(Continued)*

Keyword	Definition
normalblocklimit	<p>“12.5.7 Message Priority Based on Size” on page 387</p> <p>Forces messages above this size to nonurgent priority.</p>
parameterlengthlimit	<p>“12.9.6 Controlling the Length of General and Filename Content-type and Content-disposition Parameters” on page 414</p> <p>Controls the points at which general content-type and content-disposition parameters are truncated.</p>
recipientcutoff	<p>“12.9.7 Limiting Message Recipients” on page 414</p> <p>Rejects message if recipients exceed this value.</p>
recipientlimit	<p>“12.9.7 Limiting Message Recipients” on page 414</p> <p>Limits the number of recipient addresses that will be accepted for the message.</p>
rejectsmtplonglines	<p>“12.9.5 Handling SMTP Mail with Lines Exceeding 1000 Characters” on page 414</p> <p>Rejects messages that contain lines longer than 1000 characters (including CRLF).</p>
sourceblocklimit	<p>“12.9.2 Specifying Absolute Message Size Limits” on page 411</p> <p>Maximum number of MTA blocks allowed per incoming message.</p>
truncatesmtplonglines	<p>“12.9.5 Handling SMTP Mail with Lines Exceeding 1000 Characters” on page 414</p> <p>Truncate the line when it is over 1000 characters.</p>
wrapsmtplonglines	<p>“12.9.5 Handling SMTP Mail with Lines Exceeding 1000 Characters” on page 414</p> <p>Wrap the line when it is over 1000 characters.</p>
urgentblocklimit	<p>“12.5.7 Message Priority Based on Size” on page 387</p> <p>Forces messages above this size to normal priority.</p>

TABLE 12-16 SMTP Authentication, SASL and TLS Keywords

Keyword	Definition
(See “12.4.4 SMTP Authentication, SASL, and TLS” on page 376 for greater functional granularity)	
authrewrite	<p>“12.4.3 TCP/IP Connection and DNS Lookup Support” on page 368</p> <p>Used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers.</p>
maysaslserver	<p>“12.4.4 SMTP Authentication, SASL, and TLS” on page 376</p> <p>Permit clients to attempt to use SASL authentication.</p>
maytls	<p>“12.4.8 Transport Layer Security” on page 379</p> <p>Causes the MTA to offer TLS to incoming connections and to attempt TLS upon outgoing connections.</p>

TABLE 12-16 SMTP Authentication, SASL and TLS Keywords *(Continued)*

Keyword	Definition
maytlsclient	<p>“12.4.8 Transport Layer Security” on page 379</p> <p>The MTA SMTP client will attempt TLS use when sending outgoing messages, if sending to an SMTP server that supports TLS.</p>
maytlsserver	<p>“12.4.8 Transport Layer Security” on page 379</p> <p>The MTA SMTP server will advertise support for the STARTTLS extension and will allow TLS use when receiving messages.</p>
msexchange	<p>“12.4.7 Specifying Microsoft Exchange Gateway Channels” on page 379</p> <p>Used on TCP/IP channels to tell the MTA that this is a channel that communicates with Microsoft Exchange gateways and clients.</p>
mustsaslserver	<p>“12.4.4 SMTP Authentication, SASL, and TLS” on page 376</p> <p>SMTP server does not accept messages unless remote client successfully authenticates.</p>
musttls	<p>“12.4.8 Transport Layer Security” on page 379</p> <p>Insist upon TLS in both outgoing and incoming connections.</p>
musttlsclient	<p>“12.4.8 Transport Layer Security” on page 379</p> <p>The MTA SMTP client will insist on TLS use when sending outgoing messages (the MTA will issue the STARTTLS command and that command must succeed).</p>
musttlsserver	<p>“12.4.8 Transport Layer Security” on page 379</p> <p>The MTA SMTP server will advertise support for the STARTTLS extension and will insist upon TLS use when receiving incoming messages.</p>
nomsexchange	<p>“12.4.3 TCP/IP Connection and DNS Lookup Support” on page 368</p> <p>Default.</p>
nosasl	<p>“12.4.4 SMTP Authentication, SASL, and TLS” on page 376</p> <p>SASL authentication is not permitted or attempted.</p>
nosaslserver	<p>“12.4.4 SMTP Authentication, SASL, and TLS” on page 376</p> <p>SASL authentication is not permitted.</p>
notls	<p>“12.4.8 Transport Layer Security” on page 379</p> <p>TLS will not be permitted or attempted.</p>
notlsclient	<p>“12.4.8 Transport Layer Security” on page 379</p> <p>TLS use will not be attempted by the MTA SMTP client on outgoing connections (the STARTTLS command will not be issued during outgoing connections).</p>

TABLE 12-16 SMTP Authentication, SASL and TLS Keywords *(Continued)*

Keyword	Definition
notlsserver	<p>“12.4.8 Transport Layer Security” on page 379</p> <p>TLS use will not be permitted by the MTA SMTP server on incoming connections (the STARTTLS extension will not be advertised by the SMTP server nor the command itself accepted).</p>
saslswitchchannel	<p>“12.4.4 SMTP Authentication, SASL, and TLS” on page 376</p> <p>Cause incoming connections to be switched to a specified channel upon a client’s successful use of SASL.</p>
tlsswitchchannel	<p>“12.4.8 Transport Layer Security” on page 379</p> <p>Cause incoming connections to be switched to a specified channel upon a client’s successful TLS negotiation. It takes a required value, specifying the channel to which to switch.</p>

TABLE 12-17 SMTP Commands and Protocol Keywords

Keyword	Definition
(See “ 12.4.2 SMTP Command and Protocol Support ” on page 360 for greater functional granularity)	
allowetrn	<p>“12.4.2.3 ETRN Command Support” on page 364</p> <p>Honors ETRN commands.</p>
blocketrn	<p>“12.4.2.3 ETRN Command Support” on page 364</p> <p>Blocks ETRN commands.</p>
checkehlo	<p>“12.4.2.2 EHLO Command Support” on page 363</p> <p>Checks the SMTP response banner to determine whether to use EHLO or HELO.</p>
chunkingclient	<p>“12.4.6 Support for SMTP Chunking” on page 378</p> <p>Enable server chunking support (default).</p>
chunkingserver	<p>“12.4.6 Support for SMTP Chunking” on page 378</p> <p>Enable server chunking support (default).</p>
disableetrn	<p>“12.4.2.3 ETRN Command Support” on page 364</p> <p>Disable support for the ETRN SMTP command.</p>
domainetrn	<p>“12.4.2.3 ETRN Command Support” on page 364</p> <p>Honors only those ETRN commands that specify a domain.</p>
domainvrfy	<p>“12.4.2.4 VRFY Command Support” on page 365</p> <p>Issues VRFY commands using a full address.</p>
ehlo	<p>“12.4.2.2 EHLO Command Support” on page 363</p> <p>Uses the SMTP EHLO command on initial connections.</p>

TABLE 12-17 SMTP Commands and Protocol Keywords *(Continued)*

Keyword	Definition
eightbit	<p>“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366</p> <p>Channel supports eight-bit characters.</p>
eightnegotiate	<p>“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366</p> <p>Channel should negotiate use of eight-bit transmission if possible.</p>
eightstrict	<p>“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366</p> <p>Reject messages with headers that contain unnegotiated eight-bit data.</p>
expnallow	<p>“12.4.2.5 EXPN Support” on page 365</p> <p>Allows EXPN even if it has been disabled at the SMTP server level with the DISABLE_EXPAND SMTP channel option.</p>
expndisable	<p>“12.4.2.5 EXPN Support” on page 365</p> <p>Disables EXPN unconditionally.</p>
expndefault	<p>“12.4.2.5 EXPN Support” on page 365</p> <p>Allows EXPN if the SMTP server is set to allow it.</p>
localvrfy	<p>“12.4.2.4 VRFY Command Support” on page 365</p> <p>Issues VRFY commands using a local address.</p>
mailfromdnsverify	<p>“12.4.2.6 DNS Domain Verification” on page 366</p> <p>Verifies domain used on MAIL FROM: command exists in the DNS.</p>
nochunkingclient	<p>“12.4.6 Support for SMTP Chunking” on page 378</p> <p>Disable server chunking support.</p>
nochunkingserver	<p>“12.4.6 Support for SMTP Chunking” on page 378</p> <p>Disable server chunking support.</p>
noehlo	<p>“12.4.2.2 EHLO Command Support” on page 363</p> <p>Does not use the EHLO command.</p>
nomailfromdnsverify	<p>“12.4.2.6 DNS Domain Verification” on page 366</p> <p>Does not verify that the domain used on the MAIL FROM: command exists in the DNS.</p>
nosendetrn	<p>“12.4.2.3 ETRN Command Support” on page 364</p> <p>Does not send ETRN commands.</p>
nosmtp	<p>“12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362</p> <p>Does not support the SMTP protocol. This is the default.</p>

TABLE 12-17 SMTP Commands and Protocol Keywords *(Continued)*

Keyword	Definition
novrfy	<p>“12.4.2.4 VRFY Command Support” on page 365</p> <p>Does not issue VRFY commands.</p>
sendetrn	<p>“12.4.2.3 ETRN Command Support” on page 364</p> <p>Sends ETRN commands.</p>
sevenbit	<p>“12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366</p> <p>Do not support 8-bit characters; 8-bit characters must be encoded.</p>
silentetrn	<p>“12.4.2.3 ETRN Command Support” on page 364</p> <p>Honors ETRN commands without echoing channel information.</p>
smtp	<p>“12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362</p> <p>Supports the SMTP protocol. The keyword <code>smtp</code> is mandatory for all SMTP channels. (This keyword is equivalent to <code>smtp_crlf</code>.)</p>
smtp_cr	<p>“12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362</p> <p>Accepts lines terminated with a carriage return (CR) without a following line feed (LF).</p>
smtp_crlf	<p>“12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362</p> <p>Lines must be terminated with a carriage return (CR) line feed (LF) sequence.</p>
smtp_crclrf	<p>“12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362</p> <p>Lines may be terminated with any of a carriage return (CR), or a line feed (LF) sequence, or a full CRLF.</p>
smtp_lf	<p>“12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362</p> <p>Accepts lines terminated with linefeed (LF) without preceding CR.</p>
streaming	<p>“12.4.2.8 Protocol Streaming” on page 368</p> <p>Controls the degree of protocol streaming used in the protocol associated with a channel.</p>
vrfyallow	<p>“12.4.2.4 VRFY Command Support” on page 365</p> <p>Provides informative responses to VRFY commands.</p>
vrfydefault	<p>“12.4.2.4 VRFY Command Support” on page 365</p> <p>Provides default responses to VRFY command, according to channel's HIDE_VERIFY option setting.</p>
vrfyhide	<p>“12.4.2.4 VRFY Command Support” on page 365</p> <p>Provides obfuscatory responses to SMTP VRFY command.</p>

TABLE 12-18 TCP/IP Connection and DNS Lookup Support Keywords

Keyword	Definition
TCP/IP Connection and DNS Lookup Support (See “ 12.4.3 TCP/IP Connection and DNS Lookup Support ” on page 368 for greater functional granularity)	
cacheeverything	“12.4.3.2 Caching for Channel Connection Information” on page 371 Caches all connection information.
cachefailures	“12.4.3.2 Caching for Channel Connection Information” on page 371 Caches only connection failure information.
cachesuccesses	“12.4.3.2 Caching for Channel Connection Information” on page 371 Caches only connection success information.
connectalias	“12.6.5 Address Rewriting Upon Message Dequeue” on page 393 Deliver to whatever host is listed in the recipient address.
connectcanonical	“12.6.5 Address Rewriting Upon Message Dequeue” on page 393 Connect to the host alias for the system to which the MTA would be connected.
daemon	“12.4.3.10 Target Host Choice” on page 375 Connects to a specific host system regardless of the envelope address.
defaultmx	“12.4.3.5 TCP/IP MX Record Support” on page 373 Channel determines whether to do MX lookups from network.
defaultnameservers	“12.4.3.6 Nameserver Lookups” on page 373 Consults TCP/IP stack’s choice of nameservers.
forwardcheckdelete	“12.4.3.3 Reverse DNS Lookups” on page 371 If reverse DNS lookup performed, next performs a forward lookup on the returned name to check that the returned IP number matches the original; if not, deletes the name and use the IP address.
forwardchecknone	“12.4.3.3 Reverse DNS Lookups” on page 371 Does not perform a forward lookup after a DNS reverse lookup.
forwardchecktag	“12.4.3.3 Reverse DNS Lookups” on page 371 If a reverse DNS lookup has been performed, next performs a forward lookup on the returned name to check that the returned IP number matches the original; if not, tags the name with *.
identnone	“12.4.3.4 IDENT Lookups” on page 372 No perform IDENT lookups; performs IP to hostname translation; includes both hostname and IP address in Received: header.

TABLE 12-18 TCP/IP Connection and DNS Lookup Support Keywords *(Continued)*

Keyword	Definition
identnonelimited	<p>“12.4.3.4 IDENT Lookups” on page 372</p> <p>No IDENT lookups; does perform IP to hostname translation, but does not use the hostname during channel switching; includes both hostname and IP address in Received: header.</p>
identnonnumeric	<p>“12.4.3.4 IDENT Lookups” on page 372</p> <p>Does not perform IDENT lookups or IP to hostname translation.</p>
identnonesymbolic	<p>“12.4.3.4 IDENT Lookups” on page 372</p> <p>Does not perform IDENT lookups; does perform IP to hostname translation; includes only the hostname in Received: header.</p>
identtcp	<p>“12.4.3.4 IDENT Lookups” on page 372</p> <p>Performs IDENT lookups on incoming SMTP connections and IP to hostname translation; include both hostname and IP address in Received: header</p>
identtcplimited	<p>“12.4.3.4 IDENT Lookups” on page 372</p> <p>Performs IDENT lookups on incoming SMTP connections and IP to hostname translation, but do not use the hostname during channel switching. Include hostname and IP address in Received: header.</p>
identtcpnumeric	<p>“12.4.3.4 IDENT Lookups” on page 372</p> <p>Performs IDENT lookups on incoming SMTP connections, but does not perform IP to hostname translation.</p>
identtcpsymbolic	<p>“12.4.3.4 IDENT Lookups” on page 372</p> <p>Performs IDENT lookups on incoming SMTP connections and IP to hostname translation; includes only hostname in Received: header.</p>
interfaceaddress	<p>“12.4.3.1 TCP/IP Port Number and Interface Address” on page 370</p> <p>Binds to the specified TCP/IP interface address.</p>
lastresort	<p>“12.4.3.7 Last Resort Host” on page 374</p> <p>Specifies a last resort host.</p>
mailfromdnsverify	<p>“12.4.2.6 DNS Domain Verification” on page 366</p> <p>Verifies that the domain used on the MAIL FROM: command exists in the DNS.</p>
mx	<p>“12.4.3.5 TCP/IP MX Record Support” on page 373</p> <p>TCP/IP network and software supports MX records lookup.</p>
nameservers	<p>“12.4.3.6 Nameserver Lookups” on page 373</p> <p>Specifies a list of nameservers to consult rather than consulting the TCP/IP stack's own choice of nameservers; nameservers requires a space separated list of IP addresses for the nameservers.</p>

TABLE 12-18 TCP/IP Connection and DNS Lookup Support Keywords *(Continued)*

Keyword	Definition
nocache	<p>“12.4.3.2 Caching for Channel Connection Information” on page 371</p> <p>Does not cache any connection information.</p>
nomailfromdnsverify	<p>“12.4.2.6 DNS Domain Verification” on page 366</p> <p>Does not verify that the domain used on the MAIL FROM: command exists in the DNS.</p>
nomx	<p>“12.4.3.5 TCP/IP MX Record Support” on page 373</p> <p>TCP/IP network does not support MX lookups.</p>
nonrandommx	<p>“12.4.3.5 TCP/IP MX Record Support” on page 373</p> <p>Does MX lookups; does not randomize returned entries with equal precedence.</p>
port	<p>“12.4.3.1 TCP/IP Port Number and Interface Address” on page 370</p> <p>Specifies the default port number for SMTP connections. The standard port is 25.</p>
randommx	<p>“12.4.3.5 TCP/IP MX Record Support” on page 373</p> <p>Does MX lookups; randomizes returned entries with equal precedence.</p>
single	<p>“12.4.3.10 Target Host Choice” on page 375</p> <p>Specifies that a separate copy of the message should be created for each destination address on the channel.</p>
single_sys	<p>“12.4.3.10 Target Host Choice” on page 375</p> <p>Creates single copy of message for each destination system used.</p>
threaddepth	<p>“12.5.8 SMTP Channel Threads” on page 387</p> <p>Number of messages triggering new thread with multithreaded SMTP client.</p>

TABLE 12-19 Miscellaneous Keywords

Keyword	Definition
addresssr	<p>“15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518</p> <p>Controls SRS encoding.</p>
deferralrejectlimit	<p>“12.12.8 Setting Limits on Bad RCPT TO Addresses” on page 425</p> <p>Sets limit on the number of bad RCPT TO: addresses</p>
caption	<p>“12.12.9 Set Channel Displays for Monitoring Framework” on page 425</p> <p>Sets short channel display string for Monitoring Framework</p>

TABLE 12-19 Miscellaneous Keywords (*Continued*)

Keyword	Definition
description	<p>“12.12.9 Set Channel Displays for Monitoring Framework” on page 425</p> <p>Sets channel display string for Monitoring Framework</p>
destinationsrs	<p>“15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518</p> <p>Controls SRS encoding.</p>
dispositionchannel	<p>“12.12.1 Process Channel Overrides” on page 418</p> <p>Overrides the process channel as the place to initially queue delivery status notifications (DSNs).</p>
destinationfilter	<p>“12.12.4 Specifying Mailbox Filter File Location” on page 419</p> <p>Used on general MTA channels to specify a channel-level filter to apply to outgoing messages.</p>
filter	<p>“12.12.4 Specifying Mailbox Filter File Location” on page 419</p> <p>Takes a required URL argument describing the filter file location</p>
noaddressrs	<p>“15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518</p> <p>Controls SRS encoding.</p>
nodeestinationfilter	<p>“12.12.4 Specifying Mailbox Filter File Location” on page 419</p> <p>No channel mailbox filter is enabled for either direction of the channel.</p>
nodestinationsrs	<p>“15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518</p> <p>Controls SRS encoding.</p>
nosourcefilter	<p>“12.12.4 Specifying Mailbox Filter File Location” on page 419</p> <p>No channel mailbox filter is enabled for source channel.</p>
nosourcesrs	<p>“15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518</p> <p>Controls SRS encoding.</p>
nofilter	<p>“12.12.4 Specifying Mailbox Filter File Location” on page 419</p> <p>The default and means that a user mailbox filters are not enabled for the channel.</p>
notificationchannel	<p>“12.12.1 Process Channel Overrides” on page 418</p> <p>Overrides the process channel as the place to initially queue message disposition notifications (MDNs).</p>
sourcefilter	<p>“12.12.4 Specifying Mailbox Filter File Location” on page 419</p> <p>Used on general MTA channels to specify a channel-level filter to apply to incoming messages</p>
sourcesrs	<p>“15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518</p> <p>Controls SRS encoding.</p>

TABLE 12-19 Miscellaneous Keywords (*Continued*)

Keyword	Definition
submit	<p>“12.12.2 Channel Operation Type” on page 419</p> <p>Used to mark a channel as a submit-only channel.</p>
user	<p>“12.12.3 Pipe Channel” on page 419</p> <p>Used on pipe channels to indicate under what user name to run.</p>

12.4 Configuring SMTP Channels

Depending on the type of installation, Messaging Server provides several SMTP channels at installation time (see table below). These channels implement SMTP over TCP/IP. The multithreaded TCP SMTP channel includes a multithreaded SMTP server that runs under the control of the Dispatcher. Outgoing SMTP mail is processed by the channel program `tcp_smtp_client`, and runs as needed under the control of the Job Controller.

TABLE 12-20 SMTP Channels

Channel	Definition
<code>tcp_local</code>	Receives inbound messages from remote SMTP hosts. Depending on whether you use a smarthost/firewall configuration, either sends outbound messages directly to remote SMTP hosts or sends outbound messages to the smarthost/firewall system.
<code>tcp_intranet</code>	Receives and sends messages within the intranet.
<code>tcp_auth</code>	Used as a switch channel for <code>tcp_local</code> ; authenticated users switch to the <code>tcp_auth</code> channel to avoid relay-blocking restrictions.
<code>tcp_submit</code>	Accepts message submissions—usually from user agents—on the reserved submission port 587 (see RFC 2476).
<code>tcp_tas</code>	IA special channel used by sites doing Unified Messaging.

You can modify the definitions of these channels or create new channels by adding or removing channel keywords as described in this section. In addition, an option file may be used to control various characteristics of TCP/IP channels. Such an option file must be stored in the MTA configuration directory (`msg-svr-base/config`) and named *x_option*, where *x* is the name of the channel. Refer to the “[Option File](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for details.

This section is divided into the following subsections:

- “[12.4.1 Configuring SMTP Channel Options](#)” on page 360
- “[12.4.2 SMTP Command and Protocol Support](#)” on page 360
- “[12.4.3 TCP/IP Connection and DNS Lookup Support](#)” on page 368

- “12.4.4 SMTP Authentication, SASL, and TLS” on page 376
- “12.4.5 Using Authenticated Addresses from SMTP AUTH in Header” on page 377
- “12.4.6 Support for SMTP Chunking” on page 378
- “12.4.7 Specifying Microsoft Exchange Gateway Channels” on page 379
- “12.4.8 Transport Layer Security” on page 379

12.4.1 Configuring SMTP Channel Options

TCP/IP channel option files control various characteristics of TCP/IP channels. Channel option files must be stored in the MTA configuration directory and named *x_option*, where *x* is the name of the channel. For example, */msg-svr-base/config/tcp_local_option*

The option file consists of one or more keywords and associated values. For example you can disable mailing list expansion on your server by including the `DISABLE_EXPAND` keyword in the option file and setting the value to 1.

Other option file keywords allow you to:

- Set a limit on the number of recipients allowed per message (`ALLOW_RECIPIENTS_PER_TRANSACTION`)
- Set a limit on the number of messages allowed per connection (`ALLOW_TRANSACTIONS_PER_SESSION`)
- Fine tune the type of information logged to the MTA log file (`LOG_CONNECTION`, `LOG_TRANSPORTINFO`)
- Specify the maximum number of simultaneous outbound connections that the client channel program allows (`MAX_CLIENT_THREADS`)

SMTP channel options that take effect after channel switching should be placed in the appropriate channel option file (that is, `tcp_local_option`, `tcp_auth_option`, `tcp_intranet_option`). SMTP channel option files that take effect before channel switching MUST be placed in the channel option file for the original channel—usually `tcp_local`, because messages usually enter the MTA via the `tcp_local` channel before channel switching occurs.

For information about all channel option keywords and syntax, see the *Sun Java System Messaging Server 6.3 Administration Reference*.

12.4.2 SMTP Command and Protocol Support

You can specify whether an SMTP channel supports certain SMTP commands, such as EHLO, ETRN, EXPN and VRFY. You can also specify whether the channel support DNS domain verification, which characters the channel accepts as line terminators, and so on. This section describes the following:

- “12.4.2.1 Channel Protocol Selection and Line Terminators” on page 362
- “12.4.2.2 EHLO Command Support” on page 363
- “12.4.2.3 ETRN Command Support” on page 364
- “12.4.2.4 VRFY Command Support” on page 365
- “12.4.2.5 EXPN Support” on page 365
- “12.4.2.6 DNS Domain Verification” on page 366
- “12.4.2.7 Character Set Labeling and Eight-Bit Data” on page 366
- “12.4.2.8 Protocol Streaming” on page 368

Table 12–21 summarizes the keywords described in this section.

TABLE 12–21 SMTP Command and Protocol Keywords

Channel Keyword(s)	Description
Protocol Selection and Line Terminators	Specifies whether the channel supports the SMTP protocol and specifies the character sequences accepted as line terminators.
<code>smtp</code>	Supports the SMTP protocol. The keyword <code>smtp</code> is mandatory for all SMTP channels. (This keyword is equivalent to <code>smtp_crlf</code> .)
<code>nosmtp</code>	Does not support the SMTP protocol. This is the default.
<code>smtp_cr</code>	Accepts lines terminated with a carriage return (CR) without a following line feed (LF).
<code>smtp_crlf</code>	Lines must be terminated with a carriage return (CR) line feed (LF) sequence.
<code>smtp_lf</code>	Accepts lines terminated with a linefeed (LF) without a preceding CR.
<code>smtp_crorlf</code>	Lines may be terminated with any of a carriage return (CR), or a line feed (LF) sequence, or a full CRLF.
EHLO keywords	Specifies how the channel handles EHLO commands
<code>ehlo</code>	Uses the SMTP EHLO command on initial connections.
<code>checkehlo</code>	Checks the SMTP response banner to determine whether to use EHLO or HELO.
<code>noehlo</code>	Does not use the EHLO command.
ETRN keywords	Specifies how the channel handles ETRN commands (requests for queue processing)
<code>allowetrn</code>	Honors ETRN commands.
<code>blocketrn</code>	Blocks ETRN commands.
<code>domainetrn</code>	Honors only those ETRN commands that specify a domain.
<code>silentetrn</code>	Honors ETRN commands without echoing channel information.
<code>sendetrn</code>	Sends ETRN commands.
<code>nosendetrn</code>	Does not send ETRN commands.

TABLE 12-21 SMTP Command and Protocol Keywords *(Continued)*

Channel Keyword(s)	Description
VRFY keywords	Specifies how the channel handles VRFY commands
domainvrfy	Issues VRFY commands using a full address.
localvrfy	Issues VRFY commands using a local address.
novrfy	Does not issue VRFY commands.
vrfyallow	Provides informative responses to VRFY commands.
vrfydefault	Provides default responses to VRFY command, according to channel's HIDE_VERIFY option setting.
vrfyhide	Provides obfuscatory responses to SMTP VRFY command.
EXPN Keywords	Specifies how the channel handles EXPN keywords
expnallow	Allows EXPN even if it has been disabled at the SMTP server level with the DISABLE_EXPAND SMTP channel option.
expndisable	Disables EXPN unconditionally.
expndefault	Allows EXPN if the SMTP server is set to allow it. (Default)
DNS Domain Verification	Specifies whether the channel performs DNS domain verification
mailfromdnsverify	Verifies that the domain used on the MAIL FROM: command exists in the DNS.
nomailfromdnsverify	Does not verify that the domain used on the MAIL FROM: command exists in the DNS.
Character Sets and Eight-bit data	Specifies how the channel handles eight-bit data (Note: Although these keywords are commonly used on SMTP channels, they are potentially relevant to any sort of channel.)
charset7	Default character set to associate with 7-bit text messages
charset8	Default character set to associate with 8-bit text messages
charsetsc	Default character set to associate with 7-bit text containing the escape character
eightbit	Channel supports eight-bit characters.
eightnegotiate	Channel should negotiate use of eight-bit transmission if possible.
eightstrict	Channel should reject messages with headers that contain illegal eight-bit data.
sevenbit	Channel does not support eight-bit characters; eight-bit characters must be encoded.
Protocol streaming	Specify degree of protocol streaming for channel to use.
streaming	Controls the degree of protocol streaming used in the protocol associated with a channel.

12.4.2.1**Channel Protocol Selection and Line Terminators**Keywords: `smtp`, `nosmtp`, `smtp_crlf`, `smtp_cr`, `smtp_crorlf`, `smtp_lf`

The `smtp` and `nosmtp` keywords specify whether or not a channel supports the SMTP protocol. The `smtp` keyword, or one of its variations, is mandatory for all SMTP channels.

The keywords `smtp_crlf`, `smtp_cr`, `smtp_croRLF`, and `smtp_lf` can be used on SMTP channels to specify the character sequences that the MTA will accept as line terminators. The keyword `smtp_crlf` or `smtp` means that lines must be terminated with a carriage return (CR) line feed (LF) sequence. The keyword `smtp_lf` means that an LF without a preceding CR is accepted. The keyword `smtp_cr` means that a CR is accepted without a following LF. Finally, `smtp_croRLF` means that any of CR, LF, or the standard CRLF sequence are allowed as the SMTP line terminator. These options affect only the handling of incoming material.

Because the SMTP standard requires CRLF as the line terminator, the MTA always generates the standard CRLF sequence. The various `smtp` keywords merely control whether the MTA will accept additional non-standard line terminators. For example, you can specify `smtp_crlf` if you want the MTA to accept only strictly legal SMTP messages and reject any messages with nonstandard line terminators.

12.4.2.2

EHLO Command Support

Keywords: `ehlo`, `noehlo`, `checkehlo`

The SMTP protocol has been extended (RFC 1869) to allow for negotiation of additional commands. This is done by using the new EHLO command, which replaces RFC 821's HELO command. Extended SMTP servers respond to EHLO by providing a list of the extensions they support. Unextended servers return an unknown command error and the client then sends the old HELO command instead.

This fallback strategy normally works well with both extended and unextended servers. Problems can arise, however, with servers that do not implement SMTP according to RFC 821. In particular, some noncompliant servers are known to drop the connection on receipt of an unknown command.

The SMTP client implements a strategy whereby it attempts to reconnect and use HELO when any server drops the connection on receipt of an EHLO. However, this strategy might not work if the remote server not only drops the connection but also goes into a problematic state upon receipt of EHLO.

The channel keywords `ehlo`, `noehlo`, and `checkehlo` are provided to deal with such situations. The `ehlo` keyword tells the MTA to use the EHLO command on all initial connection attempts. The `noehlo` keyword disables all use of the EHLO command. The `checkehlo` keyword tests the response banner returned by the remote SMTP server for the string "ESMTP". If this string is found EHLO is used; if not, HELO is used. The default behavior is to use EHLO on all initial connection attempts, unless the banner line contains the string "fire away", in which case HELO is used; note that there is no keyword corresponding to this default behavior, which lies between the behaviors resulting from the `ehlo` and `checkehlo` keywords.

12.4.2.3

ETRN Command Support

Keywords: `allowetrn`, `blocketrn`, `disableetrn`, `domainetrn`, `silentetrn`, `sendetrn`, `nosendetrn`, `novrfy`

The ETRN command, defined in RFC 1985, provides an extension to the SMTP service whereby an SMTP client and server can interact to give the server an opportunity to start the processing of its queues for messages to go to a given host.

Using ETRN, an SMTP client can request that a remote SMTP server start processing the message queues destined for sending to the SMTP client. Thus, ETRN provides a way to implement “polling” of remote SMTP systems for messages incoming to one’s own system. This can be useful for systems that have only transient connections between each other, for example, sites that are set up as secondary mail exchange (MX) hosts for other sites that only have a dial-up connection to the Internet. By enabling this command, you permit remote, possibly dial-up, servers to request delivery of their mail.

The SMTP client specifies on the SMTP ETRN command line the name of the system to which to send messages (generally the SMTP client system’s own name). If the remote SMTP server supports the ETRN command, it will trigger execution of a separate process to connect back to the named system and send any messages awaiting delivery for that named system.

Responding to ETRN Commands

The `allowetrn`, `blocketrn`, `domainetrn`, and `silentetrn` keywords control the MTA response when a sending SMTP client issues the ETRN command, requesting that the MTA attempt to deliver messages in the MTA queues.

By default, the MTA will attempt to honor all ETRN commands; that is, the `allowetrn` keyword is enabled. You can specify that the MTA not honor ETRN commands by including the `blocketrn` keyword in the channel definition.

You can specify that the MTA honor all ETRN commands, but without echoing the name of the channel that the domain matched and that the MTA will be attempting to run by including the `silentetrn` keyword. The `domainetrn` keyword specifies that the MTA honor only ETRN commands that specify a domain; it also causes the MTA not to echo back the name of the channel that the domain matched and that the MTA will be attempting to run.

`disableetrn` disables support for the ETRN command entirely; ETRN is not advertised by the SMTP server as a supported command.

Sending ETRN Commands

The `sendetrn` and `nosendetrn` channel keywords control whether the MTA sends an ETRN command at the beginning of an SMTP connection. The default is `nosendetrn`, meaning that the MTA will not send an ETRN command. The `sendetrn` keyword tells the MTA to send an

ETRN command, if the remote SMTP server says it supports ETRN. The `sendetrn` keyword should be followed by the name of the system requesting that its messages receive a delivery attempt.

12.4.2.4 VRFY Command Support

Keywords: `domainvrfy`, `localvrfy`, `vrfyallow`, `vrfydefault`, `vrfyhide`

The VRFY command enables SMTP clients to send a request to an SMTP server to verify that mail for a specific user name resides on the server. The VRFY command is defined in RFC 821.

The server sends a response indicating whether the user is local or not, whether mail will be forwarded, and so on. A response of 250 indicates that the user name is local; a response of 251 indicates that the user name is not local, but the server can forward the message. The server response includes the mailbox name.

Sending a VRFY Command

Under normal circumstances there is no reason to issue a VRFY command as part of an SMTP dialogue. The SMTP RCPT TO command should perform the same function that VRFY does and return an appropriate error. However, servers exist that can accept any address in a RCPT TO (and bounce it later), whereas these same servers perform more extensive checking as part of a VRFY command.

By default, the MTA does not send a VRFY command (the `novrfy` keyword is enabled).

If necessary, the MTA can be configured to issue the SMTP VRFY command by including the `domainvrfy` or `localvrfy` keyword in the channel definition. The keyword `domainvrfy` causes a VRFY command to be issued with a full address (`user@host`) as its argument. The `localvrfy` keyword causes the MTA to issue a VRFY command with just the local part of the address (`user`).

Responding to a VRFY Command

The `vrfyallow`, `vrfydefault`, and `vrfyhide` keywords control the SMTP server's response when a sending SMTP client issues an SMTP VRFY command.

The `vrfyallow` keyword tells the MTA to issue a detailed, informative response. The `vrfydefault` tells the MTA to provide a detailed, informative response, unless the channel option `HIDE_VERIFY=1` has been specified. The `vrfyhide` keyword tells the MTA to issue only a vague, ambiguous response. These keywords allow per-channel control of VRFY responses, as opposed to the `HIDE_VERIFY` option, which normally applies to all incoming TCP/IP channels handled through the same SMTP server.

12.4.2.5 EXPN Support

Keywords: `expnallow`, `expndisable`, `expndefault`

expnallow allows EXPN even if it has been disabled at the SMTP server level with the DISABLE_EXPAND SMTP channel option. expndisable disables EXPN unconditionally. expndefault allows EXPN if the SMTP server is set to allow it (default). Expansion can be disabled on a per-list basis, but if it is disabled at the server level, the per-list settings are irrelevant.

12.4.2.6

DNS Domain Verification

Keywords: mailfromdnsverify, nomailfromdnsverify

Setting `mailfromdnsverify` on an incoming TCP/IP channel causes the MTA to verify that an entry in the DNS exists for the domain used on the SMTP MAIL FROM command and to reject the message if no such entry exists. The default, `nomailfromdnsverify`, means that no such check is performed. Note that performing DNS checks on the return address domain may result in rejecting some desired valid messages (for instance, from legitimate sites that simply have not yet registered their domain name, or at times of bad information in the DNS); it is contrary to the spirit of being generous in what you accept and getting the e-mail through, expressed in RFC 1123, Requirements for Internet Hosts. However, some sites may desire to perform such checks in cases where unsolicited bulk email (UBE) is being sent with forged e-mail addresses from non-existent domains.

Because the introduction of DNS wildcard entries in the COM and ORG top-level domains has made `mailfromdnsverify` less useful, the `mailfromdnsverify` code has been modified. When the DNS returns one or more A records, these values are compared against the domain literals specified by the new MTA option `BLOCKED_MAIL_FROM_IPS`. If a match is found, the domain is considered to be invalid. In order to restore correct behavior the current correct setting is:

`BLOCKED_MAIL_FROM_IPS=[64.94.110.11]`

This option's value defaults to an empty string.

12.4.2.7

Character Set Labeling and Eight-Bit Data

Keywords: charset7, charset8, charsetesc, sevenbit, eightbit, eightnegotiate, eightstrict

Character Set Labeling

The MIME specification provides a mechanism to label the character set used in a plain text message. Specifically, a `charset=` parameter can be specified as part of the `Content-type:` header line. Various character set names are defined in MIME, including US-ASCII (the default), ISO-8859-1, ISO-8859-2, and many more that have been subsequently defined.

Some existing systems and user agents do not provide a mechanism for generating these character set labels; as a result, some plain text messages may not be properly labeled. The `charset7`, `charset8`, and `charsetesc` channel keywords provide a per-channel mechanism to

specify character set names to be inserted into message headers which lack character set labelling. Each keyword requires a single argument giving the character set name. The names are not checked for validity. Note, however, that character set conversion can only be done on character sets specified in the character set definition file `charsets.txt` found in the MTA table directory. The names defined in this file should be used if possible.

The `charset7` character set name is used if the message contains only seven bit characters; the `charset8` character set name will be used if eight bit data is found in the message; `charsetsc` will be used if a message containing only seven bit data happens to contain escape characters also. If the appropriate keyword is not specified no character set name will be inserted into `Content-type:` header lines.

Note that the `charset8` keyword also controls the MIME encoding of 8-bit characters in message headers (where 8-bit data is unconditionally illegal). The MTA normally MIME-encodes any (illegal) 8-bit data encountered in message headers, labeling it as the UNKNOWN charset if no `charset8` value has been specified.

These character set specifications never override existing labels; that is, they have no effect if a message already has a character set label or is of a type other than text. It is usually appropriate to label MTA local channels as follows:

```
l ... charset7 US-ASCII charset8 ISO-8859-1 ...  
hostname
```

If there is no `Content-type` header in the message, it is added. This keyword also adds the `MIME-version:` header line if it is missing.

The `charsetsc` keyword tends to be particularly useful on channels that receive unlabeled messages using Japanese or Korean character sets that contain the escape character.

Eight-Bit Data

Some transports restrict the use of characters with ordinal values greater than 127 (decimal). Most notably, some SMTP servers will strip the high bit and thus garble messages that use characters in this eight-bit range.

Messaging Server provides facilities to automatically encode such messages so that troublesome eight bit characters do not appear directly in the message. This encoding can be applied to all messages enqueued to a given channel by specifying the `sevenbit` keyword. A channel should be marked `eightbit` if no such restriction exists.

The SMTP protocol disallows `eightbit` “unless the remote SMTP server explicitly says it supports the SMTP extension allowing `eightbit`.” Some transports such as extended SMTP may actually support a form of negotiation to determine if eight bit characters can be transmitted. Therefore, the use of the `eightnegotiate` keyword is strongly recommended to instruct the channel to encode messages when negotiation fails. This is the default for all channels; channels that do not support negotiation will simply assume that the transport is capable of handling eight bit data.

The `eightstrict` keyword tells Messaging Server to reject any incoming messages with headers that contain illegal eight bit data.

12.4.2.8 Protocol Streaming

Keywords: `streaming`

Some mail protocols support streaming operations. This means that the MTA can issue more than one operation at a time and wait for replies to each operation to arrive in batches. The `streaming` keyword controls the degree of protocol streaming used in the protocol associated with a channel. This keyword requires an integer parameter; how the parameter is interpreted is specific to the protocol in use.

Under normal circumstances, the extent of streaming support available is negotiated using the SMTP pipelining extension. As such this keyword should never be used under normal circumstances.

The streaming values available range from 0 to 3. A value of 0 specifies no streaming, a value of 1 causes groups of RCPT TO commands to stream, a value of 2 causes MAIL FROM/RCPT TO stream, and a value of 3 causes HELO/MAIL FROM/RCPT TO or RSET/MAIL FROM/RCPT TO streaming to be used. The default value is 0.

12.4.3 TCP/IP Connection and DNS Lookup Support

You can specify information about how the server handles TCP/IP connections and address lookups. This section describes the following:

- “[12.4.3.1 TCP/IP Port Number and Interface Address](#)” on page 370
- “[12.4.3.2 Caching for Channel Connection Information](#)” on page 371
- “[12.4.3.3 Reverse DNS Lookups](#)” on page 371
- “[12.4.3.4 IDENT Lookups](#)” on page 372
- “[12.4.3.5 TCP/IP MX Record Support](#)” on page 373
- “[12.4.3.6 Nameserver Lookups](#)” on page 373
- “[12.4.3.7 Last Resort Host](#)” on page 374
- “[12.4.3.8 Alternate Channels for Incoming Mail \(Switch Channels\)](#)” on page 374
- “[12.4.3.9 Source Channel Switching Based on User or Domain Settings](#)” on page 375
- “[12.4.3.10 Target Host Choice](#)” on page 375

Table 12–22 lists the TCP/IP connection and DNS lookup keywords described in this section.

TABLE 12-22 TCP/IP Connection and DNS Lookup Keywords

Channel Keyword(s)	Description
Port Selection and Interface Address	Specifies the default port number and interface address for SMTP connections
port	Specifies the default port number for SMTP connections. The standard port is 25.
interfaceaddress	Binds to the specified TCP/IP interface address.
Cache Keywords	Specifies how connection information is cached
cacheeverything	Caches all connection information.
cachefailures	Caches only connection failure information.
cachesuccesses	Caches only connection success information.
nocache	Does not cache any connection information.
Reverse DNS Lookups	Specifies how to handle Reverse DNS lookups on incoming SMTP connections
forwardcheckdelete	If a reverse DNS lookup has been performed, next performs a forward lookup on the returned name to check that the returned IP number matches the original; if not, deletes the name and use the IP address.
forwardchecknone	Does not perform a forward lookup after a DNS reverse lookup.
forwardchecktag	If a reverse DNS lookup has been performed, next performs a forward lookup on the returned name to check that the returned IP number matches the original; if not, tags the name with *.
IDENT Lookups/DNS Reverse Lookups	Specifies how to handle IDENT lookups and DNS Reverse Lookups on incoming SMTP connections
identnone	Does not perform IDENT lookups; does perform IP to hostname translation; includes both hostname and IP address in Received: header.
identnonelimited	Does not perform IDENT lookups; does perform IP to hostname translation, but does not use the hostname during channel switching; includes both hostname and IP address in Received: header.
identnonenumeric	Does not perform IDENT lookups or IP to hostname translation.
identnonesymbolic	Does not perform IDENT lookups; does perform IP to hostname translation; includes only the hostname in Received: header.
identtcp	Performs IDENT lookups on incoming SMTP connections and IP to hostname translation; include both hostname and IP address in Received: header
identtcplimited	Performs IDENT lookups on incoming SMTP connections and IP to hostname translation, but do not use the hostname during channel switching. Includes both hostname and IP address in Received: header.
indenttcpnumeric	Performs IDENT lookups on incoming SMTP connections, but does not perform IP to hostname translation.

TABLE 12–22 TCP/IP Connection and DNS Lookup Keywords *(Continued)*

Channel Keyword(s)	Description
<code>identtcpsymbolic</code>	Performs IDENT lookups on incoming SMTP connections and IP to hostname translation; includes only hostname in Received: header.
MX Record Support and TCP/IP Nameserver	Specifies whether and how the channel supports MX record lookups
<code>mx</code>	TCP/IP network and software supports MX records lookup.
<code>nomx</code>	TCP/IP network does not support MX lookups.
<code>defaultmx</code>	Channel determines whether to do MX lookups from network.
<code>randommx</code>	Does MX lookups; randomizes returned entries with equal precedence.
<code>nonrandommx</code>	Does MX lookups; does not randomize returned entries with equal precedence.
<code>nameservers</code>	Specifies a list of nameservers to consult rather than consulting the TCP/IP stack's own choice of nameservers; <code>nameservers</code> requires a space separated list of IP addresses for the nameservers.
<code>defaultnameservers</code>	Consults TCP/IP stack's choice of nameservers.
<code>lastresort</code>	Specifies a last resort host.
Switch keywords	Controls selection of alternate channels for incoming mail
<code>allowswitchchannel</code>	Allows switching to this channel from a <code>switchchannel</code> channel
<code>noswitchchannel</code>	Stays with the server channel; does not switch to the channel associated with the originating host; does not permit being switched to.
<code>switchchannel</code>	Switches from the server channel to the channel associated with the originating host.
<code>userswitchchannel</code>	Source channel switching based on user or domain settings.
<code>tlsswitchchannel</code>	Switches to another channel upon successful TLS negotiation.
<code>saslswitchchannel</code>	Switches to another channel when SASL authentication is successful.
Target Host Choice and Storage of Message Copies	Specifies a target host system and how message copies are stored.
<code>daemon</code>	Connects to a specific host system regardless of the envelope address.
<code>single</code>	Specifies that a separate copy of the message should be created for each destination address on the channel.
<code>single_sys</code>	Creates a single copy of the message for each destination system used.

12.4.3.1 TCP/IP Port Number and Interface Address

Keywords: `port`, `interfaceaddress`

The SMTP over TCP/IP channels normally connect to port 25 when sending messages. The `port` keyword can be used to instruct an SMTP over TCP/IP channel to connect to a nonstandard port. Note that this keyword complements the Dispatcher option `PORT`, which controls which ports the MTA listens on for accepting SMTP connections.

The `interfaceaddress` keyword controls the address to which a TCP/IP channel binds as the source address for outbound connections; that is, on a system with multiple interface addresses this keyword controls which address will be used as the source IP address when the MTA sends outgoing SMTP messages. Note that this keyword complements the Dispatcher option `INTERFACE_ADDRESS`, which controls which interface address a TCP/IP channel listens on for accepting incoming connections and messages.

12.4.3.2

Caching for Channel Connection Information

Keywords: `cacheeverything`, `nocache`, `cachefailures`, `cachesuccesses`

Channels using the SMTP protocol maintain a cache containing a history of prior connection attempts. This cache is used to avoid reconnecting multiple times to inaccessible hosts, which can waste lots of time and delay other messages. The cache is a per process cache and only persists during a single run of the outbound SMTP delivery channel.

The cache normally records both connection successes and failures. (Successful connection attempts are recorded in order to offset subsequent failures—a host that succeeded before but fails now doesn't warrant as long of a delay before making another connection attempt as does one that has never been tried or one that has failed previously.)

However, the caching strategy used by the MTA is not necessarily appropriate for all situations. Therefore channel keywords are provided to adjust the MTA cache.

The `cacheeverything` keyword enables all forms of caching and is the default. The `nocache` keyword disables all caching.

The `cachefailures` keyword enables caching of connection failures but not successes—this forces a somewhat more restricted retry than `cacheeverything` does. Finally, `cachesuccesses` caches only successes. This last keyword is effectively equivalent to `nocache` for SMTP channels.

12.4.3.3

Reverse DNS Lookups

Keywords: `forwardchecknone`, `forwardchecktag`, `forwardcheckdelete`

The `forwardchecknone`, `forwardchecktag`, and `forwardcheckdelete` channel keywords can modify the effects of doing reverse DNS lookups. These keywords can control whether the MTA does a forward lookup of an IP name found using a DNS reverse lookup, and if such forward lookups are requested, specify what the MTA does if the forward lookup of the IP name does not match the original IP number of the connection.

The `forwardchecknone` keyword is the default, and means that no forward lookup is done. The `forwardchecktag` keyword tells the MTA to do a forward lookup after each reverse lookup and to tag the IP name with an asterisk (*), if the number found using the forward lookup does not match that of the original connection. The `forwardcheckdelete` keyword tells the MTA to do a forward lookup after each reverse lookup and to ignore (delete) the reverse lookup returned name if the forward lookup of that name does not match the original connection IP address; in this case, the MTA uses the original IP address instead.

Note – Having the forward lookup not match the original IP address is normal at many sites, where a more “generic” IP name is used for several different IP addresses.

12.4.3.4

IDENT Lookups

Keywords: `identnone`, `identnonelimited`, `identnonnumeric`, `identnonesymbolic`, `identtcp`, `identtcpnumeric`, `identtcpsymbolic`, `identtcplimited`

The `IDENT` keywords control how the MTA handles connections and lookups using the `IDENT` protocol. The `IDENT` protocol is described in RFC 1413.

The `identtcp`, `identtcpsymbolic`, and `identtcpnumeric` keywords tell the MTA to perform a connection and lookup using the `IDENT` protocol. The information obtained from the `IDENT` protocol (usually the identity of the user making the SMTP connection) is inserted into the `Received:` header of the message as follows:

- `identtcp` inserts the host name corresponding to the incoming IP number, as reported from a DNS reverse lookup and the IP number itself.
- `identtcpsymbolic` inserts the host name corresponding to the incoming IP number, as reported from a DNS reverse lookup; the IP number itself is not included in the `Received:` header.
- `identtcpnumeric` inserts the actual incoming IP number—no DNS reverse lookup on the IP number is performed.

Note – The remote system must be running an `IDENT` server for the `IDENT` lookup caused by `identtcp`, `identtcpsymbolic`, or `identtcpnumeric` to be useful.

Be aware that `IDENT` query attempts may incur a performance hit. Increasingly routers will “black hole” attempted connections to ports that they don’t recognize. If this happens on an `IDENT` query, then the MTA does not hear back until the connection times out (a TCP/IP stack controlled time-out, typically on the order of a minute or two).

Another performance factor occurs when comparing `identtcp`, `identtcplimited`, or `identtcpsymbolic` to `identtcpnumeric`. The DNS reverse lookup called for with `identtcp`, `identtcplimited`, or `identtcpsymbolic` incurs some additional overhead to obtain the more user-friendly host name.

The `identnone` keyword disables IDENT lookup, but does specify IP to host name translation, and both IP number and host name will be included in the `Received:` header for the message.

The `identnonesymbolic` keyword disables IDENT lookup, but does do IP to host name translation; only the host name will be included in the `Received:` header for the message.

The `identnonenumERIC` keyword disables this IDENT lookup and inhibits the usual DNS reverse lookup translation of IP number to host name, and might result in a performance improvement at the cost of less user-friendly information in the `Received:` header. This is the default.

The `identtcplimited` and `identnonelimited` keywords have the same effect as `identtcp` and `identnone`, respectively, as far as IDENT lookups, reverse DNS lookups, and information displayed in `Received:` header. Where they differ is that with `identtcpLimited` or `identnonelimited` the IP literal address is always used as the basis for any channel switching due to use of the `switchchannel` keyword, regardless of whether the DNS reverse lookup succeeds in determining a host name.

12.4.3.5

TCP/IP MX Record Support

Keywords: `mx`, `nomx`, `defaultmx`, `randommx`, `nonrandommx`

Some TCP/IP networks support the use of MX (mail forwarding) records and some do not. Some TCP/IP channel programs can be configured not to use MX records if they are not provided by the network that the MTA system is connected to. The `mx`, `nomx`, `defaultmx`, `randommx`, `nonrandommx` keywords control MX record support.

The keyword `randommx` specifies that MX lookups should be done and MX record values of equal precedence should be processed in random order. The keyword `nonrandommx` specifies that MX lookups should be done and MX values of equal precedence should be processed in the same order in which they were received.

The `mx` keyword is currently equivalent to `nonrandommx`; it might change to be equivalent to `randommx` in a future release. The `nomx` keyword disables MX lookups. The `defaultmx` keyword specifies that `mx` should be used if the network says that MX records are supported. The keyword `defaultmx` is the default on channels that support MX lookups in any form.

12.4.3.6

Nameserver Lookups

Keywords: `nameservers`, `defaultnameservers`

When name server lookups are being performed, the `nameservers` channel keyword may be used to specify a list of name servers to consult rather than consulting the TCP/IP stack's own choice of name servers. The `nameservers` keyword requires a space separated list of IP addresses for the name servers, as shown in the following example:

```
nameservers 1.2.3.1 1.2.3.2
```

The default, `defaultnameservers`, means use the TCP/IP stack's own choice of name servers.

To prevent name server lookups on UNIX, you can modify the `nsswitch.conf` file. On NT, modify the TCP/IP configuration.

12.4.3.7 Last Resort Host

Keywords: `lastresort`

The `lastresort` keyword is used to specify a host to connect to even when all other connection attempts fail. In effect this acts as an MX record of last resort. This is only useful on SMTP channels.

The keyword requires a single parameter specifying the name of the “system of last resort.” For example:

```
tcp_local single_sys smtp mx lastresort mailhub.siroe.com  
TCP-DAEMON
```

12.4.3.8 Alternate Channels for Incoming Mail (Switch Channels)

Keywords: `switchchannel`, `allowswitchchannel`, `noswitchchannel`. See also `saslswitchchannel` on “[12.4.4 SMTP Authentication, SASL, and TLS](#)” on page 376, and `tlsswitchchannel` on “[12.4.8 Transport Layer Security](#)” on page 379 and `userswitchchannel` on “[12.4.3.9 Source Channel Switching Based on User or Domain Settings](#)” on page 375

The following keywords control selection of an alternate channel for incoming mail: `switchchannel`, `allowswitchchannel`, `noswitchchannel`.

When the MTA accepts an incoming connection from a remote system, it must choose a channel with which to associate the connection. Normally this decision is based on the transfer used; for example, an incoming SMTP over TCP/IP connection is automatically associated with the `tcp_local` channel.

This convention breaks down, however, when multiple outgoing channels with different characteristics are used to handle different systems over the same transfer. When this happens, incoming connections are not associated with the same channel as outgoing connections, and the result is that the corresponding channel characteristics are not associated with the remote system.

The `switchchannel` keyword provides a way to eliminate this difficulty. If `switchchannel` is specified on the initial channel the server uses, the IP address of the connecting (originating) host will be matched against the channel table and if it matches the source channel will change accordingly. If no IP address match is found or if a match is found that matches the original default incoming channel, the MTA may optionally try matching using the host name found by doing a DNS reverse lookup. The source channel may change to any channel marked `switchchannel` or `allowswitchchannel` (the default). The `noswitchchannel` keyword specifies that no channel switching should be done to or from the channel.

Specification of `switchchannel` on anything other than a channel that a server associates with by default has no effect. At present, `switchchannel` only affects SMTP channels, but there are actually no other channels where `switchchannel` would be reasonable.

12.4.3.9 Source Channel Switching Based on User or Domain Settings

Keywords: `userswitchchannel`. See also `switchchannel` on “[12.4.3.8 Alternate Channels for Incoming Mail \(Switch Channels\)](#)” on page 374

Source channel switching based on user or domain settings is now possible. There are three new settings involved:

1. A new channel keyword `userswitchchannel`. This keyword must be present on the initial source channel for user channel switching to occur.
2. A new MTA option `LDAP_DOMAIN_ATTR_SOURCE_CHANNEL` that specifies the name of a domain-level attribute containing the name of the channel to which to switch.
3. A new MTA option `LDAP_SOURCE_CHANNEL` that specifies the name of a user-level attribute containing the name of the channel to which to switch.

Additionally, the channel being switched to must be set to allow channel switches. That is, it cannot be marked with the `noswitchchannel` keyword. Switching is done based on information returned by rewriting the `MAIL FROM` address. Note that `MAIL FROM` addresses are easily forged so this functionality should be used with extreme care.

12.4.3.10 Target Host Choice

Keywords: `daemon`, `multiple`, `single`, `single_sys`

The interpretation and usage of the `daemon` keyword depends upon the type of channel to which it is applied.

The `daemon` keyword is used on SMTP channels to control the choice of a target host.

Normally, channels connect to whatever host is listed in the envelope address of the message being processed. The `daemon` keyword is used to tell the channel to instead connect to a specific remote system, generally a firewall or mail hub system, regardless of the envelope address. The actual remote system name should appear directly after the `daemon` keyword, as shown in the following example:

```
tcp_firewall smtp mx daemon firewall.acme.com  
TCP-DAEMON
```

If the argument after the `daemon` keyword is not a fully qualified domain name, the argument will be ignored and the channel will connect to the channel's official host. The official host is the fully qualified hostname associated with the channel. This can be specified in the second line of a three line channel block:

```
tcp_firewall smtp mx daemon router
firewall.acme.com
TCP-DAEMON
```

The official host can also be specified after TCP-DAEMON in a two-line channel block so outbound connections will identify themselves as a particular host:

```
tcp_firewall smtp mx daemon router
TCP-DAEMON firewall.acme.com
```

When specifying the firewall or gateway system name as the official host name, the argument given to the `daemon` keyword is typically specified as `router`, as shown in the following example:

```
tcp_firewall smtp mx daemon router
firewall.acme.com
TCP-DAEMON
```

Messaging Server allows multiple destination addresses to appear in each queued message. Some channel programs, however, may only be able to process messages with one recipient, or with a limited number of recipients, or with a single destination system per message copy. For example, the SMTP client programs for the TCP/IP channels only establish a connection to a single remote host in a given transaction, so only addresses to that host can be processed (this despite the fact that a single channel is typically used for all TCP/IP traffic).

Other keywords of interest are `multiple`, `single` and `single_sys`. The `multiple` keyword, the default, creates a single copy of the message for the entire channel. The `single` keyword specifies that a separate copy of the message should be created for each destination address on the channel. The `single_sys` keyword creates a single copy of the message for each destination system used. Note that at least one copy of each message is created for each channel the message is queued to, regardless of the keywords used.

12.4.4

SMTP Authentication, SASL, and TLS

Keywords: `maysaslserver`, `mustsaslserver`, `nosasl`, `nosaslserver`, `saslswitchchannel`, `nosaslswitchchannel`)

You can control whether the Messaging Server supports authentication to the SMTP server using SASL (Simple Authentication and Security Layer). SASL is defined in RFC 2222 and for more information about SASL, SMTP authentication, and security is in [Chapter 23, “Configuring Security and Access Control.”](#)

The `maysaslserver`, `mustsaslserver`, `nosasl`, `nosaslserver`, `switchchannel`, and `saslswitchchannel` channel keywords are used to configure SASL (SMTP AUTH) use during the SMTP protocol by SMTP channels such as TCP/IP channels.

`nosasl` is the default and means that SASL authentication is not permitted or attempted. It subsumes `nosaslserver`, which means that SASL authentication is not permitted. Specifying `maysaslserver` causes the SMTP server to permit clients to attempt to use SASL authentication. Specifying `mustsaslserver` causes the SMTP server to insist that clients use SASL authentication; the SMTP server does not accept messages unless the remote client successfully authenticates.

Use `saslswitchchannel` to cause incoming connections to be switched to a specified channel upon a client's successful use of SASL. It takes a required value, specifying the channel to which to switch.

12.4.5 Using Authenticated Addresses from SMTP AUTH in Header

Keywords: `authrewrite`

The `authrewrite` channel keyword and associated `AUTH_REWRITE` mapping table allows modification of header and envelope addresses using addressing information obtained from authentication operations. Specifically, SASL authentication can be configured to provide an authorized email address. Normally the `SMTP AUTH` information is used, though this may be overridden via the `FROM_ACCESS` mapping. The `authrewrite` keyword takes a required bit value, according to [Table 12-23](#).

TABLE 12-23 `authrewrite` Bit Values

Bit	Value	Description
0	1	Don't change anything (default)
1	2	Add either a <code>Sender:</code> or <code>Resent-sender:</code> header field containing the address provided by the authentication operation. The <code>Resent-</code> variant is used if other <code>resent-</code> fields are present.
2	4	Add a <code>Sender:</code> header field containing the address provided by the authentication operation.

TABLE 12-23 authrewrite Bit Values (Continued)

Bit	Value	Description
3	8	<p>Construct a probe in a mapping table called AUTH_REWRITE of the form:</p> <p><i>mail-from sender from auth-sender</i></p> <p>where <i>mail-from</i> is the envelope From: address, <i>sender</i> is the address from the Sender: or Resent-Sender: header field, <i>from</i> is the address from the From: or Resent-From: header field, and <i>auth-sender</i> is the address provided by the authentication operation.</p> <p>The result is run through the AUTH_REWRITE mapping. The mapping should return a list of items separated by vertical bars (). The items are consumed, in order, by the setting of the following flags:</p> <ul style="list-style-type: none"> \$J \$K Replace the envelope From: address for the message \$Y \$T Add an appropriate Sender: or Resent-Sender: header field. \$N Reject the message. Mapping result provides text of the error message. If no text is provided, invalid originator address used error message is displayed. \$Z Add an appropriate From: or Resent-From: header field. (Note that in general overriding a From: field is a very bad idea.) <p>The Resent- variants are used if other Resent- fields are present in the header.</p>
4	16	Apply AUTH_REWRITE mapping even when authentication has not provided an authenticated address. If the bit is clear, the mapping is only applied if an authenticated address is available.
5	32	Include the source channel at the beginning of the AUTH_REWRITE mapping probe. It is separated from the remaining information by a . If the bit is clear, the channel is not included.



Caution – The \$Z flag should be highly restricted as there are few legitimate uses for modifying envelope and header addresses.

12.4.6 Support for SMTP Chunking

Keywords: chunkingclient, chunkingserver, nochunkingclient, nochunkingserver

Support for SMTP chunking (RFC 3030) has been added to both the SMTP client and server. This support is enabled by default. Four new channel keywords can be used to control whether or not chunking is allowed. They are as follows:

- chunkingclient - Enable client chunking support (default).
- chunkingserver - Enable server chunking support (default).
- nochunkingclient - Disable client chunking support.
- nochunkingserver - Disable server chunking support.

The log file action field has been extended to indicate whether or not chunking was used to transfer a given message. Specifically, a C will be appended if chunking is used. Note that ESMTP has to be used for chunking to work, so you'll typically see field values like EEC or DEC.

12.4.7 Specifying Microsoft Exchange Gateway Channels

Keywords: `msexchange`, `nomsexchange`

The `msexchange` channel keyword may be used on TCP/IP channels to tell the MTA that this is a channel that communicates with Microsoft Exchange gateways and clients. When placed on an incoming TCP/IP channel which has SASL enabled (via a `maysaslserver` or `mustsaslserver` keyword), it causes the MTA's SMTP server to advertise AUTH using an “incorrect” format (based upon the original ESMTP AUTH specification, which was actually incompatible with correct ESMTP usage, rather than the newer, corrected AUTH specification). Some Microsoft Exchange clients, for instance, does not recognize the correct AUTH format and only recognizes the incorrect AUTH format.

The `msexchange` channel keyword also causes advertisement (and recognition) of broken TLS commands.

`nomsexchange` is the default.

12.4.8 Transport Layer Security

Keywords: `maytls`, `maytlsclient`, `maytlsserver`, `musttls`, `musttlsclient`, `musttlsserver`, `notls`, `notlsclient`, `notlsserver`, `tlsswitchchannel`

The `maytls`, `maytlsclient`, `maytlsserver`, `musttls`, `musttlsclient`, `musttlsserver`, `notls`, `notlsclient`, `notlsserver`, and `tlsswitchchannel` channel keywords are used to configure TLS use during the SMTP protocol by SMTP based channels such as TCP/IP channels.

The default is `notls`, and means that TLS will not be permitted or attempted. It subsumes the `notlsclient` keyword, which means that TLS use will not be attempted by the MTA SMTP client on outgoing connections (the STARTTLS command will not be issued during outgoing connections) and the `notlsserver` keyword, which means that TLS use will not be permitted by the MTA SMTP server on incoming connections (the STARTTLS extension will not be advertised by the SMTP server nor the command itself accepted).

Specifying `maytls` causes the MTA to offer TLS to incoming connections and to attempt TLS upon outgoing connections. It subsumes `maytlsclient`, which means that the MTA SMTP client will attempt TLS use when sending outgoing messages, if sending to an SMTP server that supports TLS, and `maytlsserver`, which means that the MTA SMTP server will advertise support for the STARTTLS extension and will allow TLS use when receiving messages.

Note that for TLS to work, the following conditions must be in place:

- Protections/ownerships of the certificates have to be set so that the `mailsrv` account can access the files.
- The directory where the certificates are stored need to have protections/ownerships set such that the `mailsrv` account can access the files within that directory.

Specifying `musttls` will cause the MTA to insist upon TLS in both outgoing and incoming connections; email will not be exchanged with remote systems that fail to successfully negotiate TLS use. It subsumes `musttlscient`, which means that the MTA SMTP client will insist on TLS use when sending outgoing messages and will not send to SMTP servers that do not successfully negotiate TLS use (the MTA will issue the STARTTLS command and that command must succeed). It also subsumes `musttlsserver`, which means that the MTA SMTP server will advertise support for the STARTTLS extension and will insist upon TLS use when receiving incoming messages and will not accept messages from clients that do not successfully negotiate TLS use.

The `tlsswitchchannel` keyword is used to cause incoming connections to be switched to a specified channel upon a client's successful TLS negotiation. It takes a required value, specifying the channel to which to switch.

12.5 Configuring Message Processing and Delivery

You can configure when the server attempts to deliver messages based on certain criteria. You can also specify parameters for job processing, such as processing limits for service jobs, or when to spawn a new SMTP channel thread. This section describes the following:

- “12.5.1 Setting Channel Directionality” on page 382
- “12.5.2 Implementing Deferred Delivery Dates” on page 382
- “12.5.3 Specifying the Retry Frequency for Messages that Failed Delivery” on page 383
- “12.5.4 Processing Pools for Channel Execution Jobs” on page 384
- “12.5.5 Service Job Limits” on page 384
- “12.5.7 Message Priority Based on Size” on page 387
- “12.5.8 SMTP Channel Threads” on page 387
- “12.5.9 Expansion of Multiple Addresses” on page 388
- “12.5.10 Enable Service Conversions” on page 389

For conceptual information on message processing and delivery, refer to [Table 12–24](#)

“12.5 Configuring Message Processing and Delivery” on page 380 summarizes the keywords described in this section.

TABLE 12-24 Message Processing and Delivery Keywords

Keyword	Definition
Immediate Delivery	Defines specification for immediate delivery of messages.
Channel Directionality	Specifies type of by which program a channel is served
bidirectional	Channel is served by a master and slave programs.
master	Channel is served by a master program (<code>master</code>).
slave	Channel is served by a slave program (<code>slave</code>).
Deferred Delivery	Defines specification for delivery of deferred jobs.
backoff	Specifies the frequency for attempted redelivery of deferred messages. Can be overridden by <code>normalbackoff</code> , <code>nonurgentbackoff</code> , <code>urgentbackoff</code> .
deferred	Implements recognition and honoring of the <code>Deferred-delivery:</code> header line.
nodeferred	Default. Specifies that <code>Deferred-delivery:</code> header line not be honored.
nonurgentbackoff	The frequency for attempted redelivery of nonurgent messages.
normalbackoff	The frequency for attempted redelivery of normal messages.
urgentbackoff	The frequency for attempted redelivery of urgent messages.
Message Priority Based on Size	Defines message priority based on message size.
nonurgentblocklimit	Forces messages above this size to lower than nonurgent priority (second class priority), meaning that the messages will always wait for the next periodic job for further processing.
normalblocklimit	Forces messages above this size to nonurgent priority.
urgentblocklimit	Forces messages above this size to normal priority.
Processing Pools for Channel Execution Jobs	Specifies the pools for processing messages of different urgencies and deferral of jobs
pool	Specifies the pool in which channels run.
after	Specifies a time delay before channels run.
Service Job Limits	Specifies the number of service jobs and the maximum number of message files to handle per job
maxjobs	Specifies the maximum number of jobs that can be running concurrently for the channel.
filesperjob	Specifies the number of queue entries to be processed by a single job.
SMTP Channel Threads	
threaddepth	Number of messages triggering new thread with multithreaded SMTP client.
Multiple Address Expansion	Defines processing for messages with many recipients

TABLE 12-24 Message Processing and Delivery Keywords *(Continued)*

Keyword	Definition
expandlimit	Processes an incoming message “off-line” when the number of addressees exceeds this limit.
expandchannel	Specifies channel in which to perform deferred expansion due to application of expandlimit.
holdlimit	Holds an incoming message when the number of addresses exceeds this limit.
Transaction Limits	Specifies connection transaction limits
transactionlimit	Limits the number of messages allowed per connection.
Undeliverable Message Notifications	Specifies when undeliverable message notifications are sent.
notices	Specifies the amount of time that may elapse before notices are sent and messages returned.
nonurgentnotices	Specifies the amount of time that may elapse before notices are sent and messages returned for messages of non-urgent priority.
normalnotices	Specifies the amount of time that may elapse before notices are sent and messages returned for messages of normal priority.
urgentnotices	Specify the amount of time which may elapse before notices are sent and messages returned for messages of urgent priority.

12.5.1 Setting Channel Directionality

Keywords: `master`, `slave`, `bidirectional`

Three keywords are used to specify whether a channel is served by a master program (`master`), a slave program (`slave`), or both (`bidirectional`). The default, if none of these keywords are specified, is `bidirectional`. These keywords determine whether the MTA initiates delivery activity when a message is queued to the channel.

The use of these keywords reflects certain fundamental characteristics of the corresponding channel program or programs. The descriptions of the various channels the MTA supports indicate when and where these keywords should be used.

12.5.2 Implementing Deferred Delivery Dates

Keywords: `deferred`, `nodeferred`

The `deferred` channel keyword implements recognition and honoring of the `Deferred-delivery:` header line. Messages with a `deferred` delivery date in the future are held in the channel queue until they either expire and are returned or the deferred delivery date is reached. See RFC 1327 for details on the format and operation of the `Deferred-delivery:` header line.

The keyword `nodeferred` is the default. It is important to realize that while support for deferred message processing is mandated by RFC 1327, actual implementation of it effectively lets people use the mail system as an extension of their disk quota and opens the system up to potential denial-of-service attacks on your disk space by arbitrary malicious users.

12.5.3 Specifying the Retry Frequency for Messages that Failed Delivery

Keywords: `backoff`, `nonurgentbackoff`, `normalbackoff`, `urgentbackoff`, `notices`

By default, the frequency of delivery retries for messages that have had delivery failures depends on the message's priority. The default intervals between delivery attempts (in minutes) is shown below. The first number after the priority indicates the number of minutes after the initial delivery failure that the first delivery retry is attempted:

```
urgent: 30, 60, 60, 120, 120, 120, 240  
normal: 60, 120, 120, 240, 240, 240, 480  
nonurgent: 120, 240, 240, 480, 480, 480, 960
```

For urgent messages, a retry is attempted 30 minutes after the initial delivery failure, 60 minutes after the first delivery retry, 60 minutes after the second retry, 120 minutes after the third and so on. Retries after the last specified attempt repeat at the same interval. Thus, for urgent messages, retries occur every 240 minutes.

Delivery attempts continue for a period of time specified by the `notices`, `nonurgentnotices`, `normalnotices`, or `urgentnotices` keywords. If a successful delivery cannot be made, a *delivery failure notification* is generated and the message is returned to sender. (For details on the `notices` keyword, see “[10.10.4.3 To Set Notification Message Delivery Intervals](#)” on page 278)

The `backoff` keywords allow you to specify customized sets of delivery retry intervals for messages of varying priorities. `nonurgentbackoff` specifies the intervals for nonurgent messages. `normalbackoff` specifies the intervals for normal messages. `urgentbackoff` specifies the intervals for urgent messages. If none of these keywords is specified, `backoff` specifies the intervals for all messages regardless of priority.

An example, is shown below:

```
urgentbackoff "pt30m" "pt1h" "pt2h" "pt3h" "pt4h" "pt5h" "pt8h" "pt16h"
```

Here, delivery retries of urgent messages is attempted 30 minutes after the initial delivery failure, one hour after the first delivery attempt (1 hour 30 minutes after initial failure), two hours after the second delivery attempt, three hours after the third, four hours after the fourth, five hours after the fifth, eight hours after the sixth, 16 hours after the seventh delivery attempt. Subsequent attempts are made every 16 hours until the period of time specified by the `notices` keyword. If a successful delivery cannot be made, a delivery failure notification is generated and

the message is returned to sender. Note that the interval syntax is in ISO 8601P and is described in the Sun Java System Messaging Server Administration Reference.

In this next example,

```
normalbackoff "pt30m" "pt1h" "pt8h" "p1d" "p2d" "p1w"
```

a delivery retry of normal messages is attempted 30 minutes after the initial delivery failure, one hour after the first delivery attempt, eight hours after the second attempt, one day after the third, two days after the fourth, one week after the fifth and repeating each week until the period of time specified by the notices keyword. If a successful delivery cannot be made, a delivery failure notification is generated and the message is returned to sender.

In this final example,

```
backoff "pt30m" "pt120m" "pt16h" "pt36h" "p3d"
```

all failed message deliveries, regardless of message priority—unless overridden by nonurgentbackoff, normalbackoff, or urgentbackoff—will be retried 30 minutes after the initial delivery failure, two hours after the first retry attempt, 16 hours after the second attempt, 36 hours after the third, three days after the fourth and repeating every three days until the period of time specified by the notices keyword. If a successful delivery cannot be made, a delivery failure notification is generated and the message is returned to sender.

12.5.4

Processing Pools for Channel Execution Jobs

Keywords: pool

You can configure various channels to share resources by running within the same pool. You might want to configure other channels to run in pools dedicated to a particular channel. Within each pool, messages are automatically sorted into different processing queues according to message priority. Higher priority messages in the pool are processed before lower-priority messages. (See “[12.5.7 Message Priority Based on Size](#)” on page 387)

The pools where the jobs are created can be selected on a channel by channel basis by using the pool keyword. The pool keyword must be followed by the name of the pool to which delivery jobs for the current channel should be pooled. The name of the pool should not contain more than twelve characters.

For further information on Job Controller concepts and configuration, refer to “[10.4.8 Job Controller File](#)” on page 256, “[10.4.8 Job Controller File](#)” on page 256, and “[12.5.5 Service Job Limits](#)” on page 384.

12.5.5

Service Job Limits

Keywords: maxjobs, filesperjob

Each time a message is enqueued to a channel, the Job Controller ensures that there is a job running to deliver the message. This might involve starting a new job process, adding a thread, or simply noting that a job is already running. A single service job may not be sufficient to ensure prompt delivery of all messages, however. (For further information on Job Controller concepts and configuration, refer to “[10.4.8 Job Controller File](#)” on page 256, “[12.5.4 Processing Pools for Channel Execution Jobs](#)” on page 384 and “[8.7 The Job Controller](#)” on page 202.)

For any given installation, there is a reasonable maximum number of processes and threads to be started for delivering messages. This maximum number depends on factors such as the number of processors, the speed of the disks, and the characteristics of the connections. In the MTA configuration, it is possible to control the following:

- The maximum number of processes to start running for a given channel (the `max_jobs` channel keyword)
- The maximum number of processes to start for a set of channels (the `JOB_LIMIT` parameter in the relevant pool section of the Job Controller configuration file)
- The number of queued messages received before a new thread or process is started (the `threaddepth` channel keyword)
- For some channels, the maximum number of threads that will run within a given delivery program (`max_client_threads` parameter in the channel option file)

The maximum number of processes to start running for a given channel is the minimum of the `max_jobs` set on the channel and the `JOB_LIMIT` set for the pool that the channel runs in.

Assume a message needs processing. In general, the Job Controller starts new processes as follows:

- If no process is running for a channel and the pool job limit has not been reached, then the Job Controller starts a new process.
- If a channel program is single-threaded or the thread limit has been reached and the backlog increases past a multiple of threads (specified with `threaddepth`) and neither the channel nor pool job limit has been reached, the Job Controller starts a new process
- If a channel program is multithreaded and the thread limit has not been reached and the backlog of messages increase past a multiple of `threaddepth`, a new thread is started.

For SMTP channels in particular, new threads or processes are started as messages are enqueued for different hosts. Thus, for SMTP channels, the Job Controller starts new processes as follows. Assume a message needs processing:

- If no process is running for an SMTP channel and the pool limit has not been reached, the Job Controller starts a new process.
- If the thread limit (`MAX_CLIENT_THREADS`) has been reached, a message is enqueued for a host not yet being serviced, and neither the channel (`max_jobs`) nor pool job limit (`JOB_LIMIT`) has been reached then a new process is started.

- If the thread limit has not been reached and a message is enqueued for a host not yet being serviced, then a new thread is started.
- If the thread limit has not been reached and a message is enqueued that takes the backlog of messages for that host increase past a multiple of `threaddepth`, a new thread is started.

See also “[12.5.8 SMTP Channel Threads](#)” on page 387.

The `filesperjob` keyword can be used to cause the MTA to create additional service jobs. This keyword takes a single positive integer parameter which specifies how many queue entries (that is, files) must be sent to the associated channel before more than one service job is created to handle them. If a value less than or equal to zero is given it is interpreted as a request to queue only one service job. Not specifying a keyword is equivalent to specifying a value of zero. The effect of this keyword is maximized; the larger number computed will be the number of service jobs that are actually created.

The `filesperjob` keyword divides the number of actual queue entries or files by the given value. Note that the number of queue entries resulting from a given message is controlled by a large number of factors, including but not limited to the use of the `single` and `single_sys` keywords and the specification of header-modifying actions in mailing lists.

The `maxjobs` keyword places an upper bound on the total number of service jobs that can be running concurrently. This keyword must be followed by an integer value; if the computed number of service jobs is greater than this value only `maxjobs` jobs will actually be created. The default for this value if `maxjobs` is not specified is 100. Normally `maxjobs` is set to a value that is less than or equal to the total number of jobs that can run simultaneously in whatever service pool or pools the channel uses.

12.5.6

Setting Connection Transaction Limits

Keywords: `transactionlimit`

`transactionlimit` limits the number of messages allowed per connection. This can be used to thwart attackers in the following way:

An attacker can connect via SMTP and send many `RCPT TO` commands in an attempt to guess legitimate email addresses. Such an attack can be thwarted by limiting the number of invalid `RCPT TO`s allowed in a transaction. The attacker may respond by using multiple transactions, but with `transactionlimit` you can limit the number of transaction allowed in an SMTP session. The attacker can use multiple sessions, but now his cost is getting prohibitive. Connection throttling can be used to limit the number of sessions in various ways making the cost really prohibitive in most cases.

This is not without cost our side, however. Some SMTP clients react badly to recipient limits, transaction limits, or both. Exceptions need to be made for these clients. But TCP channel options apply to the SMTP server unconditionally. The solution is to use channel keywords and `switchchannel` to route problematic agents to channels with larger limits.

12.5.7 Message Priority Based on Size

Keywords: `urgentblocklimit`, `normalblocklimit`, `nonurgentblocklimit`

The `urgentblocklimit`, `normalblocklimit`, and `nonurgentblocklimit` keywords may be used to instruct the MTA to downgrade the priority of messages based on size. These keywords affect the priority that the Job Controller applies when processing the message.

12.5.8 SMTP Channel Threads

Keywords: `threaddepth`,

The multithreaded SMTP client sorts outgoing messages to different destinations to different threads. The `threaddepth` keyword may be used to instruct the multithreaded SMTP client to handle only the specified number of messages in any one thread, using additional threads even for messages all to the same destination (hence normally all handled in one thread). The default for this keyword is 10.

Each time the backlog for a channel increases past a multiple of `threaddepth`, the Job Controller tries to increase the amount of processing dedicated to processing messages queued for that channel. For multithreaded channels, the Job Controller advises any job processing messages for that channel to start a new thread, or if all jobs have the maximum threads allowed for the channel (`MAX_CLIENT_THREADS` in the option for the `tcp_*` channels) it will start a new process. For single-threaded channels it will start new process. Note that the Job Controller will not start a new job if the job limit for the channel (`maxjobs`) or the pool (`JOB_LIMIT`) has been reached.

Essentially, `threaddepth` controls how aggressive jobs are scheduled. Let's consider two different situations:

- (1) a normal (outbound) SMTP channel
- (2) a forward-to-a-smart-host SMTP channel

The job controller sorts messages destined for a particular channel by the destination host, and it schedules the jobs to process messages based on the backlog on these destination hosts.

In the first instance, there will be a large number of destination hosts and the backlog for most of them will be small. There will be lots of threads running and everything should work well, except, perhaps, for destination hosts like aol, yahoo, hotmail, and so on, where there can be a large amount of traffic. With a thread depth of 128, you'd only get a second thread delivering to yahoo once the backlog reached 128. This is not desirable.

In the second instance, there is only the one destination host and having many threads delivering to that host are desirable. If anything, the default value of 10 could be too small.

Use of `threaddepth` may be of particular interest for achieving multithreading on a daemon router TCP/IP channel—a TCP/IP channel that connects to a single specific SMTP server—when the SMTP server to which the channel connects can handle multiple simultaneous connections.

12.5.9 Expansion of Multiple Addresses

Keywords: `expandlimit`, `expandchannel`, `holdlimit`

Most channels support the specification of multiple recipient addresses in the transfer of each inbound message. The specification of many recipient addresses in a single message may result in delays in message transfer processing (online delays). If the delays are long enough, network time-outs can occur, which in turn can lead to repeated message submission attempts and other problems.

The MTA provides a special facility to force deferred (offline) processing if more than a given number of addresses are specified for a single message. Deferral of message processing can decrease on-line delays enormously. Note, however, that the processing overhead is deferred, not avoided completely.

This special facility is activated by using a combination of, for instance, the generic reprocessing channel and the `expandlimit` keyword. The `expandlimit` keyword takes an integer argument that specifies how many addresses should be accepted in messages coming from the channel before deferring processing. The default value is infinite if the `expandlimit` keyword is not specified. A value of 0 will force deferred processing on all incoming addresses from the channel.

The `expandlimit` keyword must not be specified on the local channel or the reprocessing channel itself; the results of such a specification are unpredictable.

The channel actually used to perform the deferred processing may be specified using the `expandchannel` keyword; the reprocessing channel is used by default, if `expandchannel` is not specified, but use of some other reprocessing or processing channel may be useful for special purposes. If a channel for deferred processing is specified via `expandchannel`, that channel should be a reprocessing or processing channel; specification of other sorts of channels may lead to unpredictable results.

The reprocessing channel, or whatever channel is used to perform the deferred processing, must be added to the MTA configuration file in order for the `expandlimit` keyword to have any effect. If your configuration was built by the MTA configuration utility, then you should already have a reprocessing channel.

Extraordinarily large lists of recipient addresses are often a characteristic of unsolicited bulk email. The `holdlimit` keyword tells the MTA that messages coming in the channel that result in more than the specified number of recipients should be marked as .HELD messages and

enqueued to the reprocess channel (or to whatever channel is specified via the expandchannel keyword). The files will sit unprocessed in the reprocess queue awaiting manual intervention by the MTA postmaster.

12.5.10 Enable Service Conversions

Keywords: `service`, `noservice`

The `service` keyword unconditionally enables service conversions regardless of `CHARSET-CONVERSION` entry. If the `noservice` keyword is set, service conversions for messages coming into this channel must be enabled via `CHARSET-CONVERSION`.

12.6 Configuring Address Handling

This section describes keywords that deal with address handling. It consists of the following sections:

- “12.5.10 Enable Service Conversions” on page 389
- “12.6.1 Address Types and Conventions” on page 389
- “12.6.2 Interpreting Addresses that Use ! and %” on page 391
- “12.6.3 Adding Routing Information in Addresses” on page 391
- “12.6.4 Disabling Rewriting of Explicit Routing Addresses” on page 392
- “12.6.5 Address Rewriting Upon Message Dequeue” on page 393
- “12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses” on page 393
- “12.6.7 Legalizing Messages Without Recipient Header Lines” on page 394
- “12.6.8 Stripping Illegal Blank Recipient Headers” on page 395
- “12.6.9 Enabling Channel-Specific Use of the Reverse Database” on page 395
- “12.6.10 Enabling Restricted Mailbox Encoding” on page 395
- “12.6.11 Generating of Return-path Header Lines” on page 396
- “12.6.12 Constructing Received Header Lines from Envelope To and From Addresses” on page 396
- “12.6.13 Handling Comments in Address Header Lines” on page 396
- “12.6.14 Handling Personal Names in Address Header Lines” on page 397
- “12.6.15 Specifying Alias File and Alias Database Probes” on page 398
- “12.6.16 Subaddress Handling” on page 398
- “12.6.17 Enabling Channel-specific Rewrite Rules Checks” on page 399
- “12.6.18 Removing Source Routes” on page 399
- “12.6.19 Specifying Address Must be from an Alias” on page 400

12.6.1 Address Types and Conventions

Keywords: `822`, `733`, `uucp`, `header_822`, `header_733`, `header_uucp`

This group of keywords control what types of addresses the channel supports. A distinction is made between the addresses used in the transport layer (the message envelope) and those used in message headers.

12.6.1.1 822 (sourceroute)

Source route envelope addresses. This channel supports full RFC 822 format envelope addressing conventions including source routes. The keyword `sourceroute` is also available as a synonym for 822. This is the default if no other envelope address type keyword is specified.

12.6.1.2 733 (percents)

Percent sign envelope addresses. This channel supports full RFC 822 format envelope addressing with the exception of source routes; source routes should be rewritten using percent sign conventions instead. The keyword `percents` is also available as a synonym for 733.

Note – Use of 733 address conventions on an SMTP channel results in these conventions being carried over to the transport layer addresses in the SMTP envelope. This may violate RFC 821. Only use 733 address conventions when you are sure they are necessary.

12.6.1.3 uucp (bangstyle)

Bang-style envelope addresses. This channel uses addresses that conform to RFC 976 bang-style address conventions in the envelope (for example, this is a UUCP channel). The keyword `bangstyle` is also available as a synonym for `uucp`.

12.6.1.4 header_822

Source route header addresses. This channel supports full RFC 822 format header addressing conventions including source routes. This is the default if no other header address type keyword is specified.

12.6.1.5 header_733

Percent sign header addresses. This channel supports RFC 822 format header addressing with the exception of source routes; source routes should be rewritten using percent sign conventions instead.

Note – Use of 733 address conventions in message headers may violate RFC 822 and RFC 976. Only use this keyword if you are sure that the channel connects to a system that cannot deal with source route addresses.

12.6.1.6 **header_uucp**

UUCP or bang-style header addresses. The use of this keyword is not recommended. Such usage violates RFC 976.

12.6.2 **Interpreting Addresses that Use ! and %**

Keywords: `bangoverpercent`, `nobangoverpercent`, `percentonly`

Addresses are always interpreted in accordance with RFC 822 and RFC 976. However, there are ambiguities in the treatment of certain composite addresses that are not addressed by these standards. In particular, an address of the form `A!B%C` can be interpreted as either:

- A as the routing host and C as the final destination host
- or
- C as the routing host and A as the final destination host

While RFC 976 implies that mailers can interpret addresses using the latter set of conventions, it does not say that such an interpretation is required. Some situations may be better served by the former interpretation.

The `bangoverpercent` keyword forces the former `A! (B%C)` interpretation. The `nobangoverpercent` keyword forces the latter `(A!B)%C` interpretation. `nobangoverpercent` is the default.

Note – This keyword does not affect the treatment of addresses of the form `A!B%C`. These addresses are always treated as `(A!B)@C`. Such treatment is mandated by both RFC 822 and RFC 976.

The `percentonly` keyword ignores bang paths. When this keyword is set, percents are interpreted for routing.

12.6.3 **Adding Routing Information in Addresses**

Keywords: `exproute`, `noexproute`, `improute`, `noimproute`

The addressing model that the MTA deals with assumes that all systems are aware of the addresses of all other systems and how to get to them. Unfortunately, this ideal is not possible in all cases, such as when a channel connects to one or more systems that are not known to the rest of the world (for example, internal machines on a private TCP/IP network). Addresses for systems on this channel may not be legal on remote systems outside of the site. If you want to be

able to reply to such addresses, they must contain a source route that tells remote systems to route messages through the local machine. The local machine can then (automatically) route the messages to these machines.

The `exproute` keyword (short for “explicit routing”) tells the MTA that the associated channel requires explicit routing when its addresses are passed on to remote systems. If this keyword is specified on a channel, the MTA adds routing information containing the name of the local system (or the current alias for the local system) to all header addresses and all envelope `From:` addresses that match the channel. `noexproute`, the default, specifies that no routing information should be added.

The `EXPROUTE_FORWARD` option can be used to restrict the action of `exproute` to backward-pointing addresses. Another scenario occurs when the MTA connects to a system through a channel that cannot perform proper routing for itself. In this case, all addresses associated with other channels need to have routing indicated when they are used in mail sent to the channel that connects to the incapable system.

Implicit routing and the `improute` keyword is used to handle this situation. The MTA knows that all addresses matching other channels need routing when they are used in mail sent to a channel marked `improute`. The default, `noimproute`, specifies that no routing information should be added to addresses in messages going out on the specified channel. The `IMPROUTE_FORWARD` option can be used to restrict the action of `improute` to backward-pointing addresses.

The `exproute` and `improute` keywords should be used sparingly. They make addresses longer, more complex, and may defeat intelligent routing schemes used by other systems. Explicit and implicit routing should not be confused with specified routes. Specified routes are used to insert routing information from rewrite rules into addresses. This is activated by the special `A@B@C` rewrite rule template.

Specified routes, when activated, apply to all addresses, both in the header and the envelope. Specified routes are activated by particular rewrite rules and as such are usually independent of the channel currently in use. Explicit and implicit routing, on the other hand, are controlled on a per-channel basis and the route address inserted is always the local system.

12.6.4 Disabling Rewriting of Explicit Routing Addresses

Keywords: `routelocal`

The `routelocal` channel keyword causes the MTA, when rewriting an address to the channel, to attempt to “short circuit” any explicit routing in the address. Explicitly routed addresses (using `!`, `%`, or `@` characters) are simplified.

Use of this keyword on “internal” channels, such as internal TCP/IP channels, can allow simpler configuration of SMTP relay blocking.

Note that this keyword should not be used on channels that may require explicit % or other routing.

12.6.5 Address Rewriting Upon Message Dequeue

Keywords: `connectalias`, `connectcanonical`

The MTA normally rewrites addresses as it enqueues messages to its channel queues. No additional rewriting is performed during message dequeue. This presents a potential problem when host names change while there are messages in the channel queues still addressed to the old name.

The `connectalias` keyword tells the MTA to deliver to whatever host is listed in the recipient address. This is the default. The keyword `connectcanonical` tells the MTA to connect to the host alias for the system that to which the MTA would be connected.

12.6.6 Specifying a Host Name to Use When Correcting Incomplete Addresses

Keywords: `remotehost`, `noremotehost`, `defaulthost`, `nodefaulthost`

The MTA often receives addresses that do not contain domain names from misconfigured or incompliant mailers and SMTP clients. The MTA attempts to make such addresses legal before allowing them to pass further. The MTA does this by appending a domain name to the address (for example, appends `@siroe.com` to `mrochek`).

For envelope `To:` addresses missing a domain name, the MTA always assumes that the local host name should be appended. However for other addresses, such as `From:` addresses, in the case of the MTA SMTP server there are at least two reasonable choices for the domain name: the local MTA host name and the remote host name reported by the client SMTP. Or in some cases, there may be yet a third reasonable choice—a particular domain name to add to messages coming in that channel. Now, either of these two first choices are likely to be correct as both may occur operationally with some frequency. The use of the remote host's domain name is appropriate when dealing with improperly configured SMTP clients. The use of the local host's domain name may be appropriate when dealing with a lightweight remote mail client such as a POP or IMAP client that uses SMTP to post messages. Or if lightweight remote mail clients such as POP or IMAP, clients should have their own specific domain name which is not that of the local host. Then add that specific other domain name may be appropriate. The best that the MTA can do is to allow the choice to be made on a channel by channel basis.

The `noremotehost` channel keyword specifies that the local host's name should be used. The keyword `noremotehost` is the default.

The `defaulthost` channel keyword is used to specify a particular host name to append to incoming bare user id's. It must be followed by the domain name to use in completing addresses (in envelope `From:` and in headers) that come into that channel. (In the case of submit channels, the `defaulthost` keyword's first argument also affects bare envelope `To:` addresses.) An optional second domain name (that has at least one period in it) may be specified to use in completing envelope `To:` addresses. `nodefaulthost` is the default.

The `switchchannel` keyword as described, in the preceding section, “[12.4.3.8 Alternate Channels for Incoming Mail \(Switch Channels\)](#)” on page 374 associate incoming SMTP connections with a particular channel. This facility can be used to group remote mail clients on a channel where they can receive proper treatment. Alternatively, it is simpler to deploy standards-compliant remote mail clients (even if a multitude of noncompliant clients are in use) rather than attempting to fix the network-wide problem on your MTA hosts.

12.6.7 Legalizing Messages Without Recipient Header Lines

Keywords: `missingrecipientpolicy`

RFC 822 (Internet) messages are required to contain recipient header lines: `To:`, `Cc:`, or `Bcc:` header lines. A message without such header lines is illegal. Nevertheless, some broken user agents and mailers (for example, many older versions of sendmail) emit illegal messages.

The `missingrecipientpolicy` keyword takes an integer value specifying the approach to use for such messages; the default value, if the keyword is not explicitly present, is 1 (pass the illegal message through unchanged).

TABLE 12-25 `missingrecipientpolicy` Values

Value	Action
0	Use the current best practice per RFC 2822 recommendation. This is currently equivalent to value 1..
1	Pass the illegal message through unchanged.
2	Place envelope <code>To:</code> recipients in a <code>To:</code> header line.
3	Place all envelope <code>To:</code> recipients in a single <code>Bcc:</code> header line.
4	Generate a group construct (for example, “ <code>;</code> ”) <code>To:</code> header line, “ <code>To: Recipients not specified:;</code> ”
5	Generate a blank <code>Bcc:</code> header line.
6	Reject the message.

Note that the `MISSING_RECIPIENT_POLICY` option can be used to set an MTA system default for this behavior. The initial Messaging Server configuration sets `MISSING_RECIPIENT_POLICY` to 1.

12.6.8 Stripping Illegal Blank Recipient Headers

Keywords: `dropblank`, `nodropblank`

In RFC 822 (Internet) messages, any `To:`, `Resent-To:`, `Cc:`, or `Resent-Cc:` header is required to contain at least one address—such a header may not have a blank value. Nevertheless, some mailers may emit such illegal headers. The `dropblank` channel keyword, if specified on a source channel, causes the MTA to strip any such illegal blank headers from incoming messages.

12.6.9 Enabling Channel-Specific Use of the Reverse Database

Keywords: `reverse`, `noreverse`

The `reverse` keyword tells the MTA that addresses in messages queued to the channel should be checked against, and possibly modified, by the address reversal database or REVERSE mapping, if either exists. `noreverse` exempts addresses in messages queued to the channel from address reversal processing. The `reverse` keyword is the default. Refer to “[10.9 To Convert Addresses from an Internal Form to a Public Form](#)” on page 264

12.6.10 Enabling Restricted Mailbox Encoding

Keywords: `restricted`, `unrestricted`

Some mail systems have difficulty dealing with the full spectrum of addresses allowed by RFC 822. A particularly common example of this is sendmail-based mailers with incorrect configuration files. Quoted local-parts (or mailbox specifications) are a frequent source of trouble:

`"smith, ned"@siroe.com`

This is such a major source of difficulty that a methodology was laid out in RFC 1137 to work around the problem. The basic approach is to remove quoting from the address, then apply a translation that maps the characters requiring quoting into characters allowed in an atom (see RFC 822 for a definition of an atom as it is used here). For example, the preceding address would become:

`smith#m#_ned@siroe.com`

The `restricted` channel keyword tells the MTA that the channel connects to mail systems that require this encoding. The MTA then encodes quoted local-parts in both header and envelope addresses as messages are written to the channel. Incoming addresses on the channel are decoded automatically. The `unrestricted` keyword tells the MTA not to perform RFC 1137 encoding and decoding. The keyword `unrestricted` is the default.

Note – The `restricted` keyword should be applied to the channel that connects to systems unable to accept quoted local-parts. It should not be applied to the channels that actually generate the quoted local-parts. (It is assumed that a channel capable of generating such an address is also capable of handling such an address.)

12.6.11 Generating of Return-path Header Lines

Keywords: `addrreturnpath`, `noaddrreturnpath`

Normally, adding the `Return-path:` header line is the responsibility of a channel performing a final delivery. But for some channels, like the `ims-ms` channel, it is more efficient for the MTA to add the `Return-path:` header rather than allowing the channel to perform add it. The `addrreturnpath` keyword causes the MTA to add a `Return-path:` header when enqueueing to this channel.

12.6.12 Constructing Received Header Lines from Envelope To and From Addresses

Keywords: `receivedfor`, `noreceivedfor`, `receivedfrom`, `noreceivedfrom`

The `receivedfor` keyword instructs the MTA that if a message is addressed to just one envelope recipient, to include that envelope `To:` address in the `Received:` header line it constructs. The keyword `receivedfor` is the default. The `noreceivedfor` keyword instructs the MTA to construct `Received:` header lines without including any envelope addressee information.

The `receivedfrom` keyword instructs the MTA to include the original envelope `From:` address when constructing a `Received:` header line for an incoming message if the MTA has changed the envelope `From:` address due to, for example, certain sorts of mailing list expansions. `receivedfrom` is the default. The `noreceivedfrom` keyword instructs the MTA to construct `Received:` header lines without including the original envelope `From:` address.

12.6.13 Handling Comments in Address Header Lines

Keywords: `commentinc`, `commentmap`, `commentomit`, `commentstrip`, `commenttotal`, `sourcecommentinc`, `sourcecommentmap`, `sourcecommentomit`, `sourcecommentstrip`, `sourcecommenttotal`

The MTA interprets the contents of header lines only when necessary. However, all registered header lines containing addresses must be parsed to rewrite and eliminate short form addresses and otherwise convert them to legal addresses. During this process, comments (strings enclosed in parentheses) are extracted and may be modified or excluded when the header line is rebuilt.

This behavior is controlled by the use of the `commentinc`, `commentmap`, `commentomit`, `commentstrip`, and `commenttotal` keywords. The `commentinc` keyword tells the MTA to retain comments in header lines. It is the default. The keyword `commentomit` tells the MTA to remove any comments from addressing headers, for example, `To:`, `From:`, or `Cc:` header lines.

The keyword `commenttotal` tells the MTA to remove any comments from all header lines, except `Received:` header lines; this keyword is not normally useful or recommended. `commentstrip` tells the MTA to strip any nonatomic characters from all comment fields. The `commentmap` keyword runs comment strings through the `COMMENT_STRINGS` mapping table.

On source channels, this behavior is controlled by the use of the `sourcecommentinc`, `sourcecommentmap`, `sourcecommentomit`, `sourcecommentstrip`, and `sourcecommenttotal` keywords. The `sourcecommentinc` keyword indicates to the MTA to retain comments in header lines. It is the default. The `sourcecommentomit` keyword indicates to the MTA to remove any comments from addressing headers, for example, `To:`, `From:`, and `Cc:` headers. The `sourcecommenttotal` keyword indicates to the MTA to remove any comments from all headers, except `Received:` headers; as such, this keyword is not normally useful or recommended. And finally, the `sourcecommentstrip` keyword indicates to the MTA to strip any nonatomic characters from all comment fields. The `sourcecommentmap` keyword runs comment strings through source channels.

These keywords can be applied to any channel.

The syntax for the `COMMENT_STRINGS` mapping table is as follows:

(comment_text) | address

If the entry template sets the `$Y` flag, the original comment is replaced with the specified text (which should include the enclosing parentheses).

12.6.14

Handling Personal Names in Address Header Lines

Keywords: `personalinc`, `personalmap`, `personalomit`, `personalstrip`, `sourcepersonalinc`, `sourcepersonalmap`, `sourcepersonalomit`, `sourcepersonalstrip`

During the rewriting process, all header lines containing addresses must be parsed in order to rewrite and eliminate short form addresses and otherwise convert them to legal addresses.

During this process personal names (strings preceding angle-bracket-delimited addresses) are extracted and can be optionally modified or excluded when the header line is rebuilt.

This behavior is controlled by the use of the `personalinc`, `personalmap`, `personalomit`, and `personalstrip` keywords. The keyword `personalinc` tells the MTA to retain personal names in the headers. It is the default. The keyword `personalomit` tells the MTA to remove all personal names. The keyword `personalstrip` tells the MTA to strip any nonatomic characters

from all personal name fields. The `personalmap` keyword indicates to the MTA to run the personal names through the `PERSONAL_NAMES` mapping table.

On source channels, this behavior is controlled by the use of a `sourcepersonalinc`, `sourcepersonalmap`, `sourcepersonalomit`, or `sourcepersonalstrip` keyword. The `sourcepersonalinc` keyword indicates to the MTA to retain personal names in the headers. It is the default. The `sourcepersonalomit` keyword indicates to the MTA to remove all personal names. And finally, the `sourcepersonalstrip` indicates to the MTA to strip any nonatomic characters from all personal name fields. The `sourcepersonalmap` keyword indicates to the MTA to run the personal names through source channels.

These keywords can be applied to any channel. If the `PERSONAL_NAMES` mapping table returns 8-bit characters, they are UTF-8 encoded.

The syntax of the `PERSONAL_NAMES` mapping table probes is:

personal_name | *address*

If the template sets the `$Y` flag, the original personal name is replaced with the specified text.

12.6.15 Specifying Alias File and Alias Database Probes

Keywords: `aliaslocal`

Normally only addresses rewritten to the local channel (that is, the `l` channel on UNIX) are looked up in the alias file and alias database. The `aliaslocal` keyword may be placed on a channel to cause addresses rewritten to that channel to be looked up in the alias file and alias database also. The exact form of the lookup probes that are made is then controlled by the `ALIAS_DOMAINS` option.

12.6.16 Subaddress Handling

Keywords: `subaddressexact`, `subaddressrelaxed`, `subaddresswild`

As background regarding the concept of subaddresses, the native and ims-ms channels interpret a `+` character in the local portion of an address (the mailbox portion) specially: in an address of the form `name+subaddress@domain` the MTA considers the portion of the mailbox after the plus character a subaddress. The native channel treats a subaddress as additional cosmetic information and actually deliver to the account name, without regard to the subaddress; the ims-ms channel interprets the subaddress as the folder name to which to deliver.

Subaddresses also affect the lookup of aliases by the local channel (that is, the `L` channel on UNIX) and the lookup of aliases by any channel marked with the `aliaslocal` keyword, and the lookup of mailboxes by the directory channel. The exact handling of subaddresses for such

matching is configurable: when comparing an address against an entry, the MTA always first checks the entire mailbox including the subaddress for an exact match; whether or not the MTA performs additional checks after that is configurable.

The `subaddressexact` keyword instructs the MTA to perform no special subaddress handling during entry matching; the entire mailbox, including the subaddress, must match an entry in order for the alias to be considered to match. No additional comparisons (in particular, no wildcard comparisons or comparisons with the subaddress removed) are performed. The `subaddresswild` keyword instructs the MTA that after looking for an exact match including the entire subaddress, the MTA should next look for an entry of the form `name+*`. The `subaddressrelaxed` keyword instructs the MTA that after looking for an exact match and then a match of the form `name+*`, that the MTA should make one additional check for a match on just the name portion. With `subaddressrelaxed`, an alias entry of the following form matches either `name` or `name+subaddress`, transforming a plain name to `newname`, and transforming `name+subaddress` to `newname+subaddress`. The `subaddressrelaxed` keyword is the default.

```
name: newname+*
```

Thus the `subaddresswild` keyword or the `subaddressrelaxed` keyword may be useful when aliases or a directory channel are in use yet users wish to receive mail addressed using arbitrary subaddresses. These keywords obviate the need for a separate entry for every single subaddress variant on an address.

Note that these keywords only make sense for the local channel (that is, the L channel on UNIX) and the directory channel, or any channel marked with the `alias local` keyword.

Standard Messaging Server configurations relay upon the L channel indeed having `subaddressrelaxed` behavior (the default, when other keywords have not been explicitly used).

12.6.17 Enabling Channel-specific Rewrite Rules Checks

Keywords: `rules`, `norules`

The `rules` keyword tells the MTA to enforce channel-specific rewrite rule checks for this channel. This is the default. The `norules` keyword tells the MTA not to check for this channel. These two keywords are usually used for debugging and are rarely used in actual applications.

12.6.18 Removing Source Routes

Keywords: `dequeue_removeroute`

The `dequeue_removeroute` keyword removes source routes from envelope To: addresses as messages are dequeued. This keyword is currently only implemented on `tcp-*` channels. It is useful for transferring messages to systems that do not handle source routes correctly.

12.6.19 Specifying Address Must be from an Alias

Keywords: `viaaliasoptional`, `viaaliasrequired`

`viaaliasrequired` specifies that any final recipient address that matches the channel must be produced by an alias. A final recipient address refers to the match after alias expansion (if relevant) has been performed. The address cannot be handed directly to the MTA as a recipient address; that is, it is not sufficient for an address to merely rewrite to the channel. After rewriting to the channel, an address must also expand through an alias to be considered to have truly matched the channel.

The `viaaliasrequired` keyword may be used, for example, on the local channel to prevent delivery to arbitrary accounts (such as arbitrary native Berkeley mailboxes on a UNIX system).

The default is `viaaliasoptional`, which means that the final recipient addresses that match the channel are not required to be produced by an alias.

12.6.20 Recipient Address Handling

Keywords: `acceptalladdresses`, `acceptvalidaddresses`

`acceptvalidaddresses` is the default and corresponds to the MTA's standard behavior where any recipient errors are reported immediately during the SMTP dialogue. If the keyword `acceptalladdresses` is specified on a channel, then all recipient addresses are accepted during the SMTP dialogue. Any invalid addresses will have a DSN sent later.

12.7 Configuring Header Handling

This section describes keywords that deal with header and envelope information. It consists of the following sections:

- “12.7.1 Rewriting Embedded Headers” on page 401
- “12.7.2 Removing Selected Message Header Lines” on page 401
- “12.7.3 Generating/Removing X-Envelope-to Header Lines” on page 402
- “12.7.4 Converting Date to Two- or Four-Digits” on page 402
- “12.7.5 Specifying Day of Week in Date” on page 403
- “12.7.6 Automatic Splitting of Long Header Lines” on page 403
- “12.7.7 Header Alignment and Folding” on page 403
- “12.7.8 Specifying Maximum Length Header” on page 404
- “12.7.9 Sensitivity Checking” on page 404
- “12.7.10 Setting Default Language in Headers” on page 405
- “12.7.11 Controlling Message-`hash`: Headers” on page 405

12.7.1 Rewriting Embedded Headers

Keywords: `noinner`, `inner`

The contents of header lines are interpreted only when necessary. However, MIME messages can contain multiple sets of message headers as a result of the ability to imbed messages within messages (message/RFC822). The MTA normally only interprets and rewrites the outermost set of message headers. The MTA can optionally be told to apply header rewriting to inner headers within the message as well.

This behavior is controlled by the use of the `noinner` and `inner` keywords. The keyword `noinner` tells the MTA not to rewrite inner message header lines. It is the default. The keyword `inner` tells the MTA to parse messages and rewrite inner headers. These keywords can be applied to any channel.

12.7.2 Removing Selected Message Header Lines

Keywords: `headertrim`, `noheadertrim`, `headerread`, `noheaderread`, `innertrim`, `noinnertrim`

The MTA provides per-channel facilities for trimming or removing selected message header lines from messages. This is done through a combination of a channel keyword and an associated header option file or two. Header option file format is described in “[Header Option Files](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

The `headertrim` keyword instructs the MTA to consult a header option file associated with the channel and to trim the headers on messages queued to that destination channel accordingly, *after the original message headers are processed*. The `noheadertrim` keyword bypasses header trimming. The keyword `noheadertrim` is the default.

The `innertrim` keyword instructs the MTA to perform header trimming on inner message parts, that is, embedded MESSAGE/RFC822 parts, as well. The `noinnertrim` keyword, which is the default, tells the MTA not to perform any header trimming on inner message parts.

The `headerread` keyword instructs the MTA to consult a header option file associated with the channel and to trim the headers on messages enqueued by that source channel accordingly, *before the original message headers are processed*. Note that `headertrim` header trimming, on the other hand, is applied after the messages have been processed and is the destination channel, rather than the source channel. The `noheaderread` keyword bypasses message enqueue header trimming. `noheaderread` is the default.

Unlike the `headeromit` and `headerbottom` keywords, the `headertrim` and `headerread` keywords may be applied to any channel whatsoever. Note, however, that stripping away vital header information from messages may cause improper operation of the MTA. Be extremely careful when selecting headers to remove or limit. This facility exists because there are occasional situations where selected header lines must be removed or otherwise limited.



Caution – Stripping away header information from messages may cause improper operation of the MTA. Be careful when selecting headers to remove or limit. These keywords are provided for the rare situations where selected header lines must be removed or limited. Before trimming or removing any header line, you must understand the usage of that header line and have considered the possible implications of its removal.

Header options files for the `header trim` and `inner trim` keywords have names of the form `channel_headers.opt` with `channel`, the name of the channel with which the header option file is associated. Similarly, header options files for the `header read` keyword have names of the form `channel_read_headers.opt`. These files are stored in the MTA configuration directory, `instance_root/config/`.

12.7.3 Generating/Removing X-Envelope-to Header Lines

Keywords: `x_env_to`, `nox_env_to`

The `x_env_to` and `nox_env_to` keywords control the generation or suppression of X-Envelope-to header lines on copies of messages queued to a specific channel. On channels that are marked with the `single` keyword, the `x_env_to` keyword enables generation of these headers while the `nox_env_to` removes such headers from enqueued messages. The default is `nox_env_to`.

Although it will rarely make sense to do so, the `x_env_to` keyword can now be used without also setting `single` on a channel.

12.7.4 Converting Date to Two- or Four-Digits

Keywords: `datefour`, `datetwo`

The original RFC 822 specification called for two-digit years in the date fields in message headers. This was later changed to four digits by RFC 1123. However, some older mail systems cannot accommodate four-digit dates. In addition, some newer mail systems can no longer tolerate two-digit dates.

Note – Systems that cannot handle both formats are in violation of the standards.

The `datefour` and `datetwo` keywords control the MTA's processing of the year field in message header dates. The keyword `datefour`, the default, instructs the MTA to expand all year fields to four digits. Two-digit dates with a value less than 50 have 2000 added, while values greater than 50 have 1900 added.



Caution – The keyword `datetwo` instructs the MTA to remove the leading two digits from four-digit dates. This is intended to provide compatibility with incompliant mail systems that require two digit dates; it should never be used for any other purpose.

12.7.5 Specifying Day of Week in Date

Keywords: `dayofweek`, `nodayofweek`

The RFC 822 specification allows for a leading day of the week specification in the date fields in message headers. However, some systems cannot accommodate day of the week information. This makes some systems reluctant to include this information, even though it is quite useful information to have in the headers.

The `dayofweek` and `nodayofweek` keywords control the MTA's processing of day of the week information. The keyword `dayofweek`, the default, instructs the MTA to retain any day of the week information and to add this information to date and time headers if it is missing.



Caution – The keyword `nodayofweek` instructs the MTA to remove any leading day of the week information from date and time headers. This is intended to provide compatibility with incompliant mail systems that cannot process this information properly; it should never be used for any other purpose.

12.7.6 Automatic Splitting of Long Header Lines

Keywords: `maxheaderaddrs`, `maxheaderchars`

Some message transfers, notably some sendmail implementations, cannot process long header lines properly. This often leads not just to damaged headers but to erroneous message rejection. Although this is a gross violation of standards, it is nevertheless a common problem.

The MTA provides per-channel facilities to split (break) long header lines into multiple, independent header lines. The `maxheaderaddrs` keyword controls how many addresses can appear on a single line. The `maxheaderchars` keyword controls how many characters can appear on a single line. Both keywords require a single integer parameter that specifies the associated limit. By default, no limit is imposed on the length of a header line nor on the number of addresses that can appear.

12.7.7 Header Alignment and Folding

Keywords: `headerlabelalign`, `headerlinelength`

The `headerlabelalign` keyword controls the alignment point for message headers enqueued on this channel; it takes an integer-valued argument. The alignment point is the margin where the contents of headers are aligned. For example, sample header lines with an alignment point of 10 might look like this:

```
To: joe@siroe.com
From: mary@siroe.com
Subject: Alignment test
```

The default `headerlabelalign` is 0, which causes headers not to be aligned. The `headerlinelength` keyword controls the length of message header lines enqueued on this channel. Lines longer than this are folded in accordance with RFC 822 folding rules.

These keywords only control the format of the headers of the message in the message queue; the actual display of headers is normally controlled by the user agent. In addition, headers are routinely reformatted as they are transferred across the Internet, so these keywords may have no visible effect even when used in conjunction with simple user agents that do not reformat message headers.

12.7.8 Specifying Maximum Length Header

Keywords: `maxprocchars`

Processing of long header lines containing lots of addresses can consume significant system resources. The `maxprocchars` keyword is used to specify the maximum length header that the MTA can process and rewrite. Messages with headers longer than this are still accepted and delivered; the only difference is that the long header lines are not rewritten in any way. A single integer argument is required. The default is processing headers of any length.

12.7.9 Sensitivity Checking

Keywords: `sensitivitynormal`, `sensitivitypersonal`, `sensitivityprivate`
`sensitivitycompanyconfidential`

The sensitivity checking keywords set an upper limit on the sensitivity of messages that can be accepted by a channel. The default is `sensitivitycompanyconfidential`; messages of any sensitivity are allowed through. A message with no `Sensitivity:` header is considered to be of `normal`, that is, the lowest, sensitivity. Messages with a higher sensitivity than that specified by such a keyword is rejected when enqueued to the channel with an error message:

```
message too sensitive for one or more paths used
```

Note that the MTA does this sort of sensitivity checking at a per-message, not per-recipient, level: if a destination channel for one recipient fails the sensitivity check, then the message bounces for all recipients, not just for those recipients associated with the sensitive channel.

12.7.10 Setting Default Language in Headers

Keywords: language

Encoded words in headers can have a specific language. The language keyword specifies the default language.

12.7.11 Controlling Message-hash: Headers

Keywords: generatemessagehash, keepmessagehash, deletemessagehash

These keywords control Message-hash: headers on messages. Generatemessage specified on a destination channel, causes a Message-hash: header field to be inserted into the message.

Keepmessagehash will cause any existing Message-hash: field to be retained.

Deletemessagehash will delete any existing Message-hash: field. Deletemessagehash is the default.

The value placed in Message-Hash: fields is a hash of the message. Several new MTA options control how the hash is generated:

MESSAGE_HASH_ALGORITHM - The hash algorithm. Can be any of the following: md2, md4, md5 (the default), sha1, md128 (for RIPE-MD128), or md160 (for RIPE-MD160).

MESSAGE_HASH_FIELDS - Comma separated list of fields from the header to hash (in order). Any known header field can be specified. If this option is not specified it defaults to "message-id,from,to,cc,bcc, resent-message-id,resent-from,resent-to,resent-cc,resent-bcc, subject,content-id,content-type,content-description."

12.8 Attachments and MIME Processing

This section describes keywords that deal with attachments and MIME processing. It consists of the following sections:

- “12.8.1 Ignoring the Encoding Header Line” on page 406
- “12.8.2 Automatic Defragmentation of Message/Partial Messages” on page 406
- “12.8.3 Automatic Fragmentation of Large Messages” on page 408
- “12.8.4 Imposing Message Line Length Restrictions” on page 409
- “12.8.5 Interpreting Content-transfer-encoding Fields on Multiparts and Message/RFC822 Parts” on page 410

12.8.1 Ignoring the Encoding Header Line

Keywords: ignoreencoding, interpretencoding

The MTA can convert various nonstandard message formats to MIME using the Yes CHARSET-CONVERSION. In particular, the RFC 1154 format uses a nonstandard Encoding: header line. However, some gateways emit incorrect information on this header line, with the result that sometimes it is desirable to ignore this header line. The ignoreencoding keyword instructs the MTA to ignore any Encoding: header line.

Note – Unless the MTA has a CHARSET-CONVERSION enabled, such headers are ignored in any case. The interpretencoding keyword instructs the MTA to pay attention to any Encoding: header line, if otherwise configured to do so, and is the default.

12.8.2 Automatic Defragmentation of Message/Partial Messages

Keywords: defragment, nodefragment

The MIME standard provides the message/partial content type for breaking up messages into smaller parts. This is useful when messages have to traverse networks with size limits, or traverse unreliable networks where message fragmentation can provide a form of “checkpointing,” allowing for less subsequent duplication of effort when network failures occur during message transfer. Information is included in each part so that the message can be automatically reassembled after it arrives at its destination.

The defragment channel keyword and the defragmentation channel provide the means to reassemble messages in the MTA. When a channel is marked defragment, any partial messages queued to the channel are placed in the defragmentation channel queue instead. After all the parts have arrived, the message is rebuilt and sent on its way. The nodefragment disables this special processing. The keyword nodefragment is the default.

12.8.2.1 Defragmentation Channel

Messages are routed to the defragment channel if the defragment keyword is on a destination channel. That is, when the defragment keyword is present on a destination channel to which the MTA would normally enqueue a message, the MTA “looks inside” (MIME parses) the message structure and if the MTA sees that the structure is a MIME message fragment, then the MTA automatically routes the message to the defragment channel rather than straight to the (normal) destination channel.

The defragment database contains information about message fragments coming into the MTA including information indicating the host on which each message fragment is received. Once an

initial fragment has been received and noted in the defragment database, any other parts of the message that are received on any other systems using the same defragment database will get routed to the host that received the very first part. For instance:

1. `message/partial; id=123; part=x` arrives on host 1 and is routed to the defragment channel on host 1 due to the `defragment` keyword being present on what would otherwise be the destination/outbound channel.
2. Defragment channel on host 1 checks the defragment database for whether any other parts of this message have yet arrived. If none have, the defragment channel (on host 1) enters this part into the defragment database marking the part as being on host 1.
3. `message/partial; id=123; part=y` arrives on host 2 and is routed to the defragment channel on host 2 due to the `defragment` keyword being present on what would otherwise be the destination/outbound channel.
4. Defragment channel on host 2 checks the defragment database, sees that part x of this message is already present and stored on host 1. The defragment channel redirects the message fragment to host 1 (source routes the address with `@host1`).
5. `message/partial' id=123; part=y` arrives on host 1, is routed to the defragment channel, the defragment channel runs and enters it into the database, and so on.

All remaining parts of a fragmented message are redirected to the host that received the first (first receiving, not necessarily `part=1`) part of the message. They are reassembled by that host's defragment channel and eventually a reassembled, defragmented message (or if the defragmentation efforts times out, due to exceeding notices, the individual fragments are sent on as-is) is sent onwards to the real destination channel. You get some load-balancing of the defragmentation of messages depending upon which host happens to receive the "first" part of each message.

12.8.2.2

Defragmentation Channel Retention Time

Messages are retained in the defragment channel queue only for a limited time. When one half of the time before the first nondelivery notice is sent has elapsed, the various parts of a message will be sent on without being reassembled. This choice of time value eliminates the possibility of a nondelivery notification being sent about a message in the defragment channel queue.

The `notices` channel keyword controls the amount of time that can elapse before nondelivery notifications are sent, and hence also controls the amount of time messages are retained before being sent on in pieces. Set the `notices` keyword value to twice the amount of time you wish to retain messages for possible defragmentation. For example, a `notices` value of 4 would cause retention of message fragments for two days:

```
defragment notices 4
DEFFRAGMENT-DAEMON
```

12.8.2.3 Using NFS-based File Systems for Defragmentation and Vacation Caching

NFS-based file systems are often used for defragmentation and vacation caching. One application is to share the defragmentation database between the multiple MTA systems by having them all share the same defragment cache. This is done by making a link from the `msg-svr-base/config/defragment_cache` on each system to the file on which you wish to have be the shared defragment database, which will be on the shared NFS disk.

In any case, NFS servers that support proper NFS file semantics (specifically those that honor lock requests, like Solaris NFS) can be used for vacation and defragmentation caches. If NFS is used, use a soft mount option. (A hard mount is the default.) Setting a relatively short timeout value controlled by the `mount_timeo` option (see the `mount_nfs(1M)` man page) is also a good idea.

With an NFS hard mount and NFS going down, what one would see is the defragment channels on the various systems hanging. With a soft mount, the defragment channels wouldn't hang, but since they would be failing to open the defragment cache, they wouldn't be able to coordinate with defragment channels on other hosts. In the unlikely case that all of a message's fragments happened to first arrive at the same host, then that host's defragment channel would presumably be able to reassemble the message and send it onwards, properly reassembled. More likely, fragments would be on different hosts, would never get reassembled, and would get sent onwards as separate fragments once a relevant defragment channel's retention time expired.

12.8.3 Automatic Fragmentation of Large Messages

Keywords: `maxblocks`, `maxlines`

Some email systems or network transfers cannot handle messages that exceed certain size limits. The MTA provides facilities to impose such limits on a channel-by-channel basis. Messages larger than the set limits are automatically split (fragmented) into multiple, smaller messages. The content type used for such fragments is `message/partial`, and a unique ID parameter is added so that parts of the same message can be associated with one another and, possibly, be automatically reassembled by the receiving mailer.

The `maxblocks` and `maxlines` keywords are used to impose size limits beyond which automatic fragmentation are activated. Both of these keywords must be followed by a single integer value. The keyword `maxblocks` specifies the maximum number of blocks allowed in a message. An MTA block is normally 1024 bytes; this can be changed with the `BLOCK_SIZE` option in the MTA option file. The keyword `maxlines` specifies the maximum number of lines allowed in a message. These two limits can be imposed simultaneously if necessary.

Message headers are, to a certain extent, included in the size of a message. Because message headers cannot be split into multiple messages, and yet they themselves can exceed the specified

size limits, a rather complex mechanism is used to account for message header sizes. This logic is controlled by the `MAX_HEADER_BLOCK_USE` and `MAX_HEADER_LINE_USE` options in the MTA option file.

`MAX_HEADER_BLOCK_USE` is used to specify a real number between 0 and 1. The default value is 0.5. A message's header is allowed to occupy this much of the total number of blocks a message can consume (specified by the `maxblocks` keyword). If the message header is larger, the MTA takes the product of `MAX_HEADER_BLOCK_USE` and `maxblocks` as the size of the header (the header size is taken to be the smaller of the actual header size and `maxblocks`) * `MAX_HEADER_BLOCK_USE`.

For example, if `maxblocks` is 10 and `MAX_HEADER_BLOCK_USE` is the default, 0.5, any message header larger than 5 blocks is treated as a 5-block header, and if the message is 5 or fewer blocks in size it is not fragmented. A value of 0 causes headers to be effectively ignored insofar as message-size limits are concerned.

A value of 1 allows headers to use up all of the size that's available. Each fragment always contains at least one message line, regardless of whether or not the limits are exceeded by this. `MAX_HEADER_LINE_USE` operates in a similar fashion in conjunction with the `maxlines` keyword.

12.8.4 Imposing Message Line Length Restrictions

Keywords: `linelength`

The SMTP specification allows for lines of text containing up to 1000 bytes. However, some transfers may impose more severe restrictions on line length. The `linelength` keyword provides a mechanism for limiting the maximum permissible message line length on a channel-by-channel basis. Messages queued to a given channel with lines longer than the limit specified for that channel are automatically encoded.

The various encodings available in the MTA always result in a reduction of line length to fewer than 80 characters. The original message may be recovered after such encoding is done by applying an appropriate decoding filter.

Note – Encoding can only reduce line lengths to fewer than 80 characters. Specification of line length values less than 80 may not actually produce lines with lengths that comply with the stated restriction.

The `linelength` keyword causes encoding of data to perform “soft” line wrapping for transport purposes. The encoding is normally decoded at the receiving side so that the original “long” lines are recovered. For “hard” line wrapping, see the “Record, text” in [Table 13–7](#).

12.8.5 Interpreting Content-transfer-encoding Fields on Multiparts and Message/rfc822 Parts

Keywords: `interpretmultipartencoding`, `ignoremultipartencoding`,
`interpretmessageencoding`, `ignoremessageencoding`

The MIME specification prohibits the use of a content-transfer-encoding other than 7-bit, 8-bit, and binary on multipart or message/rfc822 parts. It has long been the case that some agents violate the specification and encode multiparts and message/rfc822 objects. Accordingly, the MTA has code to accept such encodings and remove them. However, recently a different standards violation has shown up, one where a content-transfer-encoding field is present with a value of quoted-printable or base63, but the part isn't actually encoded. If the MTA tries to decode such a message the result is typically a blank messages, which is pretty much what you'd expect.

Messages with this problem have become sufficiently prevalent that two new pairs of channel keywords have been added to deal with the problem - interpretation of content-transfer-encoding fields on multiparts and message/rfc822 parts can be enabled or disabled. The first pair is `interpretmultipartencoding` and `ignoremultipartencoding` and the second is `interpretmessageencoding` and `ignoremessageencoding`. The defaults are `interpretmultipartencoding` and `interpretmessageencoding`.

12.9 Limits on Messages, Quotas, Recipients, and Authentication Attempts

This section describes keywords that set size limits on messages, user quotas, and privileges. It consists of the following sections:

- “12.9.1 Limits on Unsuccessful Authentication Attempts” on page 410
- “12.9.2 Specifying Absolute Message Size Limits” on page 411
- “12.9.3 Retargeting Messages Exceeding Limit on Size or Recipients” on page 412
- “12.9.4 Handling Mail Delivery to Over Quota Users” on page 414
- “12.9.5 Handling SMTP Mail with Lines Exceeding 1000 Characters” on page 414
- “12.9.6 Controlling the Length of General and Filename Content-type and Content-disposition Parameters” on page 414
- “12.9.7 Limiting Message Recipients” on page 414
- “12.9.8 Limiting Header Size” on page 415

12.9.1 Limits on Unsuccessful Authentication Attempts

Keywords: `disconnectbadauthlimit`

This keyword can be used to place a limit on the number of unsuccessful authentication attempts that will be allowed in a session before the session is disconnected. The default value for this option is 3.

12.9.2

Specifying Absolute Message Size Limits

Keywords: `blocklimit`, `noblocklimit`, `linelimit`, `nolinelimit`, `sourceblocklimit`

Although fragmentation can automatically break messages into smaller pieces, it is appropriate in some cases to reject messages larger than some administratively defined limit, (for example, to avoid service denial attacks).

The `blocklimit`, `linelimit`, and `sourceblocklimit` keywords are used to impose absolute size limits. Each of these keywords must be followed by a single integer value.

The keyword `blocklimit` specifies the maximum number of blocks allowed in a message. The MTA rejects attempts to queue messages containing more blocks than this to the channel. An MTA block is normally 1024 bytes; this can be changed with the `BLOCK_SIZE` option in the MTA option file.

The keyword `sourceblocklimit` specifies the maximum number of blocks allowed in an incoming message. The MTA rejects attempts to submit a message containing more blocks than this to the channel. In other words, `blocklimit` applies to destination channels; `sourceblocklimit` applies to source channels. An MTA block is normally 1024 bytes; this can be changed with the `BLOCK_SIZE` option in the MTA option file.

Source block limits can also be specified on a per sender basis by specifying a user LDAP attribute with the MTA option `LDAP_SOURCEBLOCKLIMIT`, and adding this attribute to the senders LDAP entry. Source block limits are also supported based on the sender's domain. Specify a domain LDAP attribute with the MTA option `LDAP_DOMAIN_ATTR_SOURCEBLOCKLIMIT`, and adding this attribute to the sender's domain LDAP entry. There are no defaults for either of these values.

The keyword `linelimit` specifies the maximum number of lines allowed in a message. The MTA rejects attempts to queue messages containing more than this number of lines to the channel. The keywords, `blocklimit` and `linelimit`, can be imposed simultaneously, if necessary.

The MTA options `LINE_LIMIT` and `BLOCK_LIMIT` can be used to impose similar limits on all channels. These limits have the advantage that they apply across all channels. Therefore, the MTA servers can make them known to mail clients prior to obtaining message recipient information. This simplifies the process of message rejection in some protocols.

The `nolinelimit` and `noblocklimit` channel keywords are the default and mean that no limits are imposed, other than any global limits imposed via the `LINE_LIMIT` or `BLOCK_LIMIT` MTA options.

12.9.3 Retargeting Messages Exceeding Limit on Size or Recipients

Keywords: `alternatechannel`, `alternateblocklimit`, `alternatelinelimit`, `alternaterecipientlimit`

The MTA provides the ability to retarget messages that exceed a specified limit on the number of recipients, message size, or message lines to an alternate destination channel. This is implemented as a set of the following channel keywords `alternatechannel`, `alternateblocklimit`, `alternatelinelimit`, and `alternaterecipientlimit` that can be placed on any destination channel. The `alternatechannel` keyword takes a single argument specifying the name of the alternate channel to use. The other keywords each accept an integer argument specifying a corresponding threshold. A message that exceeds any of these thresholds will be enqueued to the alternate channel instead of the original destination channel.

In the following channel block example, large messages over 5,000 blocks, that would have gone out the `tcp_local` channel to the Internet, instead go out the `tcp_big` channel:

```
tcp_local smtp ... other keywords... alternatechannel tcp_big alternateblocklimit 5  
tcp-daemon
```

```
tcp_big smtp ...rest of keywords...  
tcp-big-daemon
```

```
tcp_local smtp ...other keywords... alternatechannel tcp_big alternateblocklimit 5  
tcp-daemon
```

```
tcp_big smtp ...rest of keywords...  
tcp-big-daemon
```

Here are some examples of how the `alternate*` channel keywords can be used:

- If you want to deliver large messages at a delayed or an off-hours time, you can control when the `alternatechannel` (for example, `tcp_big`) runs.

One method is to use the `imsimta qm` utility's `STOP channel_name` and `START channel_name` commands, executing these commands periodically via your own custom periodic job that is run by the Job Controller or via a cron job.

- When you want the Job Controller to process large messages or messages with many recipients in their own pool, you might also use the `alternatechannel`.

You can separate small messages or messages with few recipients from the large messages or messages with many recipients, since the latter might take longer for remote SMTP servers to process and accept; you might not want the larger messages to delay delivery of the smaller messages.

Note that the Job Controller's regular scheduling of messages and assigning of messages to threads and processes are acceptable in most configurations.

- When you want to set special TCP/IP channel time-out values for large messages or for messages with many recipients, you can use the `alternatechannel`.

In particular, setting special TCP/IP channel time-out values can be helpful if you want to send messages to remote hosts that take exceptionally long to receive large messages or messages with many recipients.

Note that the default automatic time-out adjustment should be sufficient for most configurations. At most, you might want to adjust the values from the defaults and not use a special channel. In particular, see the channel options `STATUS_DATA_RECV_PER_ADDR_TIME` and `STATUS_DATA_RECV_PER_BLOCK_TIME` in the [Sun Java System Messaging Server 6.3 Administration Reference](#).

- When you want special MIME message fragmentation for especially large messages, you can use the `alternatechannel` and the `alternateblocklimit` channel keywords along with the `maxblocks` channel keyword.

Typically, you would put the desired `maxblocks` size on your regular outbound TCP/IP channels, when you want to fragment messages over a specified size. The `maxblocks` channel keyword is normally both the threshold at which to perform fragmentation and the size to make the fragments.

But, if you want to have a larger threshold trigger and make smaller actual fragments, you can use the `alternatechannel` and `alternateblocklimit` on the outbound TCP/IP channel. You can then use the `maxblock` size on your alternate channel to fragment messages over a particular size.

- You might use the `alternatechannel` in conjunction with special filtering. For instance, a message with many recipients might need more careful scrutiny of its content in case it is spam. You might want to do different filtering based on the outgoing channel (See the `destinationfilter` channel keyword in [“12.12.4 Specifying Mailbox Filter File Location” on page 419](#)).

If you are performing relatively resource-intensive scanning (such as virus filtering) via the conversion channel, very large messages might have a resource issue. You might want to use an alternate conversion channel. Or, you might want to do special conversion procedures within the regular conversion channel, based on the outgoing channel.

- You can use the `alternatechannel` when you want large outgoing messages to go out their own channel, so that they stand out when you analyze the `mail.log*` file or in counters displays.

Furthermore, if you are trying to do careful analysis of delivery statistics, it is useful to process large messages in their own channel. This is because large messages or messages with many recipients that are sent to remote SMTP hosts are likely to take longer to finish processing, thus creating different delivery statistics for larger messages than for typical messages.

12.9.4 Handling Mail Delivery to Over Quota Users

Keywords: holdxquota, noexquota

The noexquota and holdxquota keywords control the handling of messages addressed to Berkeley mailbox users (UNIX), that is, users delivered to uid the native channel, who have exceeded their disk quotas.

noexquota tells the MTA to return messages addressed to over quota users to the message's sender. holdxquota tells the MTA to hold messages to over quota users; such messages remain in the MTA queue until they can either be delivered or they time out and are returned to their sender by the message return job.

12.9.5 Handling SMTP Mail with Lines Exceeding 1000 Characters

Keywords: rejectsmtplonglines, wrapsmtplonglines, truncatesmtplonglines

rejectsmtplonglines adds the option of rejecting messages that contain lines longer than the 1000 characters (including CRLF) that SMTP allows. The other options in this area are wrapsmtplonglines, which wraps overly long lines, and the default truncatesmtplonglines, which truncates overly long lines. Both of these keywords must be applied to the initial channel used for submission (such as `tcp_local`). It will not affect any channel that is switched to subsequently.

12.9.6 Controlling the Length of General and Filename Content-type and Content-disposition Parameters

Keywords: parameterlengthlimit and nameparameterlengthlimit

parameterlengthlimit controls the points at which general content-type and content-disposition parameters are truncated. It defaults to 1024. nameparameterlengthlimit controls the points at which the name content-type and the filename content-disposition parameters are truncated. It defaults to 128. Note that only the outermost message header is processed unless MIME processing is being performed on the message. MIME processing can be enabled in a variety of ways including, but not limited to, the `inner` keyword or the use of charset conversions.

12.9.7 Limiting Message Recipients

Keywords: recipientlimit and recipientcutoff

`recipientlimit` specifies the total number of recipient addresses that will be accepted for the message. `recipientcutoff` compares the total number of recipients that were presented to the MTA to the specified value. No message will be accepted for delivery if the limit if the value is exceeded. Both keywords accept a single integer argument. The default for both infinite unless the corresponding channel keyword is specified.

Recipient limits can also be set on a sender or sender's domain. This is done by specifying a user or domain LDAP attribute with the appropriate MTA option: `LDAP_RECIPIENTLIMIT`, `LDAP_RECIPIENTCUTOFF`, `LDAP_DOMAIN_ATTR_RECIPIENTLIMIT`, `LDAP_DOMAIN_ATTR_RECIPIENTCUTOFF`, and adding the attribute to the sender's user entry or domain entry.

12.9.8 Limiting Header Size

Keywords: `headerlimit`

Imposes a limit on the maximum size of the primary (outermost) message header. The primary message headers are silently truncated when the limit is reached. If the global MTA option, `HEADER_LIMIT`, is set, it overrides this channel-level limit. Default is no limit.

12.10 File Creation in the MTA Queue

This section describes keywords that allow you to control disk resources by specifying file creation in the MTA queue. It consists of the following sections:

- “[12.10.1 Controlling How Multiple Addresses on a Message are Handled](#)” on page 415
- “[12.10.2 Spreading a Channel Message Queue Across Multiple Subdirectories](#)” on page 416
- “[12.10.3 Setting Session Limits](#)” on page 416

12.10.1 Controlling How Multiple Addresses on a Message are Handled

Keywords: `multiple`, `addrsperfile`, `single`, `single_sys`

The MTA allows multiple destination addresses to appear in each queued message. Some channel programs may only be able to process messages with one recipient, or with a limited number of recipients, or with a single destination system per message copy. For example, the SMTP channels master program establishes a connection only to a single remote host in a given transaction, so only addresses to that host can be processed (this, despite the fact, that a single channel is typically used for all SMTP traffic).

Another example is that some SMTP servers may impose a limit on the number of recipients they can handle at one time, and they may not be able to handle this type of error.

The keywords `multiple`, `addrsperfile`, `single`, and `single_sys` can be used to control how multiple addresses are handled. The keyword `single` means that a separate copy of the message should be created for each destination address on the channel. Use of `single` on `tcp_*` channels is not recommended because it changes the way the job controller manages traffic and is not appropriate for normal SMTP scenarios. The keyword `single_sys` creates a single copy of the message for each destination system used. The keyword `multiple`, the default, creates a single copy of the message for the entire channel.

Note – At least one copy of each message is created for each channel the message is queued to, regardless of the keywords used.

The `addrsperfile` keyword is used to put a limit on the maximum number of recipients that can be associated with a single message file in a channel queue, thus limiting the number of recipients that are processed in a single operation. This keyword requires a single-integer argument specifying the maximum number of recipient addresses allowed in a message file; if this number is reached, the MTA automatically creates additional message files to accommodate them. (The default `multiple` keyword corresponds in general to imposing no limit on the number of recipients in a message file, however the SMTP channel defaults to 99.)

12.10.2 Spreading a Channel Message Queue Across Multiple Subdirectories

Keywords: `subdirs`

By default, all messages queued to a channel are stored as files in the directory `msg_svr_base/queue/channel-name`, where `channel-name` is the name of the channel. However, a channel that handles a large number of messages and tends to build up a large store of message files waiting for processing, for example, a TCP/IP channel, may get better performance out of the file system if those message files are spread across a number of subdirectories. The `subdirs` channel keyword provides this capability: it should be followed by an integer that specifies the number of subdirectories across which to spread messages for the channel, for example:

```
tcp_local single_sys smtp subdirs 10
```

12.10.3 Setting Session Limits

Keywords: `disconnectbadcommandlimit`, `disconnectrecipientlimit`, `disconnectrejectlimit`, `disconnecttransactionlimit`

Four new channel keywords provide the ability to cause the SMTP server to disconnect from the client after some number of errors have been detected:

`disconnectrecipientlimit` - Limits the number of session recipients.

`disconnectrejectlimit` - Limits the number of rejected recipients.

`disconnecttransactionlimit` - Limits the number of transactions.

`disconnectbadcommandlimit` - Limits the number of bad commands.

These are all session limits. With the exception of `disconnectbadcommandlimit`, all of these limits are checked when a MAIL FROM or RSET command is issued. If any of them have been exceeded, the server will issue a 4xy error and disconnect. The bad command limit differs only in being checked when a bad command is issued.

12.11 Configuring Logging and Debugging

This section describe logging and debugging keywords.

- “[12.11.1 Logging Keywords](#)” on page 417
- “[12.11.2 Debugging Keywords](#)” on page 417
- “[12.11.3 Setting Loopcheck](#)” on page 418

12.11.1 Logging Keywords

Keywords: `logging`, `nologging`, `logheader`

The MTA provides facilities for logging each message as it is enqueued and dequeued. The `logging` and `nologging` keywords control logging for messages on a per-channel basis. By default, the initial configuration turns on logging for all channels. You can disable logging for a particular channel by substituting the `nologging` keyword in the channel definition.

`logheader` overrides the `LOG_HEADER` MTA option on a per channel basis. A value of 0 (the default) disables message header logging. See the “[Option File](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for more information.

For more information about logging, see [Chapter 25, “Managing Logging.”](#)

12.11.2 Debugging Keywords

Keywords: `master_debug`, `slave_debug`, `nomaster_debug`, `noslave_debug`

Some channel programs include optional code to assist in debugging by producing additional diagnostic output. Two channel keywords are provided to enable generation of this debugging output on a per-channel basis. The keywords are `master_debug`, which enables debugging output in master programs, and `slave_debug`, which enables debugging output in slave programs. Both types of debugging output are disabled by default, corresponding to `nomaster_debug` and `noslave_debug`.

When activated, debugging output ends up in the log file associated with the channel program. The location of the log file may vary from program to program. Log files are usually kept in the log directory. Master programs usually have log file names of the form *x_master.log*, where *x* is the name of the channel. Slave programs usually have log file names of the form *x_slave.log*.

On UNIX, when `master_debug` and `slave_debug` are enabled for the `l` channel, users then receive `imta_sendmail.log-uniqueid` files in their current directory (if they have write access to the directory; otherwise, the debug output goes to `stdout`.) containing MTA debug information.

12.11.3 Setting Loopcheck

Keywords: `loopcheck`, `noloopcheck`

The `loopcheck` keyword places a string into the SMTP EHLO response banner in order for the MTA to check if it is communicating with itself. When `loopcheck` is set, the SMTP server advertises an XLOOP extension.

When it communicates with an SMTP server supporting XLOOP, the MTA's SMTP client compares the advertised string with the value of its MTA and immediately bounce the message if the client is in fact communicating with the SMTP server.

12.12 Miscellaneous Keywords

This section describes miscellaneous keywords. It consists of the following sections:

- “12.12.1 Process Channel Overrides” on page 418
- “12.12.2 Channel Operation Type” on page 419
- “12.12.3 Pipe Channel” on page 419
- “12.12.4 Specifying Mailbox Filter File Location” on page 419
- “12.12.5 Spam Filter Keywords” on page 420
- “12.12.6 Routing After Address Validation But Before Expansion” on page 421
- “12.12.7 NO-SOLICIT SMTP Extension Support” on page 424
- “12.12.8 Setting Limits on Bad RCPT TO Addresses” on page 425
- “12.12.9 Set Channel Displays for Monitoring Framework” on page 425

12.12.1 Process Channel Overrides

Keywords: `notificationchannel`, `dispositionchannel`

These keywords override the process channel as the place to initially queue delivery status notifications (DSNs) and message disposition notifications (MDNs), respectively. If the named channel does not exist, Messaging Server resumes using the process channel.

`notificationchannel` overrides the process channel as the place to initially queue delivery status notifications (DSNs). If the named channel does not exist, Messaging Server resumes using the process channel.

`dispositionchannel` overrides the process channel as the place to initially queue message disposition notifications (MDNs). If the named channel does not exist, Messaging Server resumes using the process channel.

12.12.2 Channel Operation Type

Keywords: `submit`

Messaging Server supports RFC 2476's Message Submission protocol. The `submit` keyword may be used to mark a channel as a submit-only channel. This is normally useful mostly on TCP/IP channels, such as an SMTP server run on a special port used solely for submitting messages; RFC 2476 establishes port 587 for such message submission use.

12.12.3 Pipe Channel

Keywords: `user`

The `user` keyword is used on pipe channels to indicate under what user name to run.

Note that the argument to `user` is normally forced to lowercase, but original case is preserved if the argument is quoted.

12.12.4 Specifying Mailbox Filter File Location

Keywords: `filter`, `nofilter`, `chanelfilter`, `nochanelfilter`, `destinationfilter`, `nodeestinationfilter`, `sourcefilter`, `nosourcefilter`, `fileinto`, `nofileinto`

The `filter` keyword may be used on the native and ims-ms channels to specify the location of user filter files for that channel. It takes a required URL argument describing the filter file location. `nofilter` is the default and means that a user mailbox filters are not enabled for the channel.

The `sourcefilter` and `destinationfilter` keywords may be used on general MTA channels to specify a channel-level filter to apply to incoming and outgoing messages, respectively. These keywords take a required URL argument describing the channel filter file location. `nosourcefilter` and `nodeestinationfilter` are the defaults and mean that no channel mailbox filter is enabled for either direction of the channel.

The obsolete `chanelfilter` and `nochanelfilter` keywords are synonyms for `destinationfilter` and `nodeestinationfilter`, respectively.

The `fileinto` keyword, currently supported only for the `ims-ms` and LMTP channels, specifies how to alter an address when a mailbox filter `fileinto` operator is applied. For `ims-ms` channels, the usual usage is:

```
fileinto $U+$S@$D
```

The above specifies that the folder name should be inserted as a sub-address into the original address, replacing any originally present sub-address.

For LMTP channels, the usual usage is:

```
fileinto @$4O:$U+$S@$D
```

where `$40` is a 4 and the letter O, not the number zero.

Note that for `fileinto` to work with LMTP, the `tcp_lmtppss` channel on the backend message store requires the `flagtransfer` keyword. (see [“16.5.1 To Configure Back End Stores with LMTP and a Minimal MTA” on page 527](#)).

12.12.5 Spam Filter Keywords

Keywords: `destinationspamfilterXoptin`, `sourcespamfilterXoptin`,
`disabledestinationspamfilterX`, `disablesourcespamfilterX`

`destinationspamfilterXoptin` specifies that all messages destined to this channel are run through filtering software X even if those services are not specified by user or domain with the `LDAP_OPTINX` LDAP attributes. (Filtering software X is defined by `spamfilterX_library` in `option.dat`.) The optin parameters follow the keyword and the available parameters depend on the filtering program. For example, Brightmail parameters are normally `spam` or `virus` or `spam.virus`. The `SpamAssassin` parameter is `spam`.

`sourcespamfilterXoptin` specifies that all messages originating from this channel are run through filtering software X. (Filtering software X is defined by `spamfilterX_library` in `option.dat`.) The system-wide default parameters follow the keyword and the available parameters depend on the filtering program. If `switchchannel` is in effect, this keyword is placed on the switched-to channel.

`sourcespamfilterX` and `destinationspamfilterX` do the same thing as `destinationspamfilterXoptin` and `destinationspamfilterXoptin`, but don't accept optin parameters. They are used with filtering software that do not pass parameters and are simply enabled or not.

`disabledestinationspamfilterX` disables spam filter X for messages destined to this channel. If a message came from a channel that enabled spam filter X (example: `destinationspamfilterXoptin`) or the filter was enabled through the use of optin attributes in a user or domain LDAP entry, this keyword will disable it.

`disablesourcespamfilterX` disables spam filter X for messages coming from this channel. If a message is sent to a channel that enables spam filter X (example: `destinationspamfilterXoptin`) or the filter was enabled through the use of optin attributes in a user or domain LDAP entry, this keyword will disable it.

For complete details on how these keywords are used, see “[To Specify Channel-level Filtering](#)” on page 469.

12.12.6 Routing After Address Validation But Before Expansion

Keywords: `aliasdetourhost`, `aliasoptindetourhost`

`aliasdetourhost` and `aliasoptindetourhost` allow source-channel-specific overriding of a hosted user’s `mailHost` attribute value. In particular, `aliasdetourhost` is commonly used to achieve a “detour” in the routing of messages destined for local (hosted on this system) users to a separate host for some kind of processing. A message can be verified (the address is a legitimate local address) on the original host, detoured to the processing host, and then returned to the original host for expansion and delivery. (Note that when we mention `aliasdetourhost` we are also describing `aliasoptindetourhost`, which is similar function to `aliasdetourhost` except detouring only occurs if the user has opted in via an LDAP following attribute.)

`aliasdetourhost` allows better configuration and use of “intermediate filtering” sorts of channels and third party filtering hosts. `aliasdetourhost` is usually used in addition to use of an alternate conversion channel. `aliasdetourhost` is used to affect the routing for the local (hosted on this system) users, while an alternate conversion channel is used to affect the routing for remote recipients.

The argument for `aliasdetourhost` is a host or domain name, or a host/domain specification. (Note that rewrite rules can handle host names, IP literal addresses, and channel tags, which are implicitly considered to be host names.) If specified on a source channel, the keyword causes alias expansion of addresses stored in LDAP to stop just prior to the point where `mailhost` information is checked (after conversion tag information has been processed). At that point the message will be sent to the `aliasdetourhost` value and the address is treated has successfully completed, but before alias expansion and after address validation.

An example of where `aliasdetourhost` can be used to circumvent various problems associated with conversion channel filtering is as follows: assume a system is set up with a front-end MTA and a back-end mail store. User have their delivery options set to `forward` and `mailbox`. The MTA uses the alternate conversion channel for anti-virus/spam system. When a message arrives for this user, the MTA alias expands and produces two recipients, one local and one remote. The remote recipient’s copy gets sent directly. The local recipient’s copy, on the other hand, goes to your conversion channel, is scanned, and returns. Alias expansion is then applied

a second time, producing a second copy to the remote recipient and the local recipient's copy is delivered normally. Net result: Two copies to the remote recipient, one to the local recipient.

Instead of using an alternate conversion channel for locally-hosted users (but possibly still using an alternate conversion channel for other recipients), a channel using `aliasdetourhost` can do the following:

- Accept messages.
- Route messages to an external spam/virus filter
- Re-accept messages for address expansion and delivery.

Example 1:

Assume a third-party scanner running on a separate host from the MTA. The following example allows user entry forwarding to be used without creating spurious duplicates while preserving the ability to perform recipient address validation prior to accepting the message.

1. Create a new channel `tcp_scanner`.

Put the `daemon` keyword on that channel, pointing to your filtering system. Add the `enqueue_removeroute` to this channel, too. The `tcp_scanner` channel looks like this in `imta.cnf`:

```
tcp_scanner smtp mx single_sys subdirs 20 noreverse maxjobs 7
pool SMTP_POOL daemon my_a-v_filter.siroe.com enqueue_removeroute
tcp_scanner-daemon
```

2. Add `aliasDetourHost` `tcp_scanner-daemon` to `tcp_local` on all inbound source `tcp` channels that you want scanned, which would likely include `tcp_local`, `tcp_submit`, `tcp_intranet` and `tcp_auth`. Here we show an example for `tcp_local` and `tcp_submit`.

```
! tcp_local
tcp_local smtp mx single_sys remotehost inner switchchannel
identnonnumeric subdirs 20 maxjobs 7 pool SMTP_POOL maytlsserver
maysaslserver saslswitchchannel tcp_auth missingrecipientpolicy 0
aliasdetourhost tcp_scanner-daemon
tcp-daemon

! tcp_submit
tcp_submit submit smtp mx single_sys mustsaslserver maytlsserver
missingrecipientpolicy 4 aliasdetourhost tcp_scanner-daemon
tcp_submit-daemon
```

Note that the argument for the `aliasdetourhost` (`tcp_scanner-daemon`) is the official hostname of the new channel `tcp_scanner`.

3. Create a rewrite rule to receive the message back from the scanning system via the `tcp_scanner` channel.

```
[1.2.3.4] $E$R$U[1.2.3.4]@tcp_scanner-daemon
```

where 1.2.3.4 is the ip address of the scanner system.

If you don't have this rewrite rule, the message will come in via one of the other `tcp*` source channels and the message will get scanned again because they all have the `aliasdetourhost`. A loop will occur.

4. Recompile your configuration, and restart dispatcher.

```
#imsimta cnbuild
#imsimta restart dispatcher
```

Example 2:

Assume a third-party scanner running on the same host as the MTA but listening on a different port. Assume mail is accepted on port 10024 and relays back on 10025.

1. Create a new channel `tcp_scanner`.

```
! tcp_scanner
tcp_scanner smtp nomx single_sys identnonnumeric subdirs 20 maxjobs
7 pool SCAN_POOL daemon 127.0.0.1 port 10024 enqueue_removeroute
tcp_scanner-daemon
```

2. Add `aliasDetourHost tcp_scanner-daemon` to `tcp_local` on all inbound source `tcp` channels that you want scanned, which would likely include `tcp_local`, `tcp_submit`, `tcp_intranet`, and so forth. Here we show an example for `tcp_local` and `tcp_submit`.

```
! tcp_local
tcp_local smtp mx single_sys remotehost inner switchchannel
identnonnumeric subdirs 20 maxjobs 7 pool SMTP_POOL maytlsserver
maysaslserverasaslswitchchannel tcp_auth missingrecipientpolicy 0
aliasdetourhost tcp_scanner-daemon
tcp-daemon
```

```
! tcp_submit
tcp_submit submit smtp mx single_sys mustsaslserver maytlsserver
missingrecipientpolicy 4 aliasdetourhost tcp_scanner-daemon
tcp_submit-daemon
```

3. Add to the mappings file to re-route outbound mail through the `tcp_scanner` channel.

CONVERSIONS

<code>in-chan=tcp_scanner;out-chan=*</code> ;CONVERT	No
<code>in-chan=tcp_*;out-chan=tcp_local</code> ;CONVERT	Yes, Channel=tcp_scanner

4. In `job_controller.cnf`, under `SMTP_POOL`, add a limit to the number of concurrent scans.

Although the scanning software should also have a limit, it's good to keep this set the same so messaging server doesn't try to send mail to the scanner when it's not going to accept the message.

```
!
[POOL=SCAN_POOL]
job_limit=2
!
```

5. Add a new service to `dispatcher.cnf` to accept mail back from the scanner on a special port and source it from `tcp_scan` as to not scan it again

```
!
[SERVICE=SMTP_SCANNING]
INTERFACE_ADDRESS=127.0.0.1
PORT=10025
IMAGE=IMTA_BIN:tcp_smtp_server
LOGFILE=IMTA_LOG:tcp_smtp_server.log
STACKSIZE=2048000
PARAMETER=CHANNEL=tcp_scanner
!
```

6. Recompile your configuration, and restart `dispatcher`.

```
# imsimta cnbuild
# imsimta restart job_controller
# imsimta restart dispatcher
```

12.12.6.1 aliasoptindetourhost

Per-user `aliasdetourhost` is now possible through the following set of features:

- `aliasoptindetourhost` channel keyword. This is similar in function to `aliasdetourhost` except detouring only occurs if the user has opted in via the following attribute. The keyword's value is a comma-separated list of potential detour hosts.
- `LDAP_DETOURHOST_OPTIN` MTA option which specifies the name of an attribute used to opt the user in to the detour (assuming of course the source channel has `aliasoptindetourhost` set). If the values of this attribute contain periods they will be compared against the list of potential detour hosts and the first host on the list that matches will be the chosen detour. If the value doesn't contain a period the first detour host will be used unconditionally.
- `ALIASDETOURHOST_NULL_OPTIN` MTA option, which is similar to `SPAMFILTERX_NULL_OPTIN` (see [Table 14-1](#)). It specifies a special value which if used in the `LDAP_DETOURHOST_OPTIN` attribute is treated as the same as the attribute being omitted. The default value is "", which means that an empty attribute value is ignored.

12.12.7 NO-SOLICIT SMTP Extension Support

Keywords: `sourcenosolicit` and `destinationnosolicit`

The NO-SOLICIT SMTP extension described in the Internet-Draft [draft-malamud-no-soliciting-07.txt](#) has been implemented in Messaging Server as a proposed standard. The following channel keywords can be used to control this facility:

`sourcenosolicit` specifies a comma-separated list of solicitation field values that will be blocked in mail submitted by this channel. This list of values will appear in the NO-SOLICIT EHLO response. Glob-style wildcards can be used in the values, however, values containing wildcards will not appear in the EHLO announcement.

`destinationnosolicit` specifies a comma-separated list of solicitation field values that will not be accepted in mail queued to this channel.

12.12.8 Setting Limits on Bad RCPT TO Addresses

Keywords: `deferralrejectlimit`

Sets a limit on the number of bad RCPT TO: addresses that are allowed during a single session. After the specified number of To: addresses have been rejected, all subsequent recipients, good or bad, are rejected with a 4xx error. Provides same functionality as the `ALLOW_REJECTIONS_BEFORE_DEFERRAL` SMTP channel keyword, but on a per-channel basis.

12.12.9 Set Channel Displays for Monitoring Framework

Keywords: `caption` and `description`

These keywords take a quoted string as an argument and are used for channel displays in the Monitoring Framework Console. If no caption or description are present, the Monitoring Framework agent invents one from the channel name.

Using Predefined Channels

When you first install Messaging Server, several channels are already defined (see [Table 13–1](#)). This chapter describes how to use pre-defined channel definitions in the MTA.

If you have not already read [Chapter 10, “About MTA Services and Configuration,”](#) you should do so before reading this chapter. For information about configuring the rewrite rules in the `imta.cnf` file, see [Chapter 11, “Configuring Rewrite Rules.”](#)

This chapter contains the following sections:

- “[13.1 Predefined Channels](#)” on page 427
- “[13.2 To Deliver Messages to Programs Using the Pipe Channel](#)” on page 429
- “[13.3 To Configure the Native \(/var/mail\) Channel](#)” on page 430
- “[13.4 To Temporarily Hold Messages Using the Hold Channel](#)” on page 431
- “[13.5 The Conversion Channel](#)” on page 431
- “[13.6 Character Set Conversion and Message Reformatting](#)” on page 452

The `defaults` Channel is described in “[12.1 Configuring Channel Defaults](#)” on page 318.

13.1 Predefined Channels

The table below lists some of the predefined channels.

TABLE 13–1 Predefined Channels

Channel	Definition
<code>defaults</code>	Used to specify which keywords are defaults for various channels. See “ 12.1 Configuring Channel Defaults ” on page 318.
<code>l</code>	UNIX only. Used to make routing decisions and for submitting mail using UNIX mail tools.

TABLE 13-1 Predefined Channels *(Continued)*

Channel	Definition
ims-ms	Performs final delivery of mail to the local store.
native	UNIX only. Delivers mail to /var/mail. (Note that Messaging Server does not support /var/mail access. User must use UNIX tools to access mail from the /var/mail store.)
pipe	Used to perform delivery via a site-supplied program or script. Commands executed by the pipe channel are controlled by the administrator via the imsmta program interface.
reprocessprocess	These channels are used for deferred, offline message processing. The reprocess channel is normally invisible as a source or destination channel; the process channel is visible like other MTA channels.
defragment	Provides the means to reassemble MIME fragmented messages.
conversion	Performs body-part-by-body-part conversions on messages flowing through the MTA.
bitbucket	Used for messages that need to be discarded.
inactive/deleted	Used to process messages for users who have been marked as inactive/deleted in the directory. Typically, bounces the message and returns custom bounce message to the sender of the message.
hold	Used to hold messages for users. For example, when a user is migrated from one mail server to another.
sms	Provides support for one-way email to an SMS gateway.
tcp_local	Implements SMTP over TCP/IP. The multithreaded TCP SMTP channel includes a multithreaded SMTP server that runs under the control of the Dispatcher. Outgoing SMTP mail is processed by the channel program tcp_smtp_client, and runs as needed under the control of the Job Controller.
tcp_intranet	tcp_local receives inbound messages from remote SMTP hosts. Depending on whether you use a smarthost/firewall configuration, either sends outbound messages directly to remote SMTP hosts or sends outbound messages to the smarthost/firewall system. Sometimes tcp_local gets mail from remote SMTP hosts via proxy or firewall. tcp_local is also sometimes used for internal relay activities.
tcp_auth	tcp_intranet receives and sends messages within the intranet.
tcp_submit	tcp_auth is used as a switch channel for tcp_local; authenticated users switch to the tcp_auth channel to avoid relay-blocking restrictions.
tcp_tas	tcp_submit accepts message submissions—usually from user agents—on the reserved submission port 587 (see RFC 2476).
	tcp_tas is a special channel used by sites doing Unified Messaging.

13.2 To Deliver Messages to Programs Using the Pipe Channel

Users might want incoming mail passed to a program instead of to their mailbox. For example, users might want their incoming mail sent to a mail sorting program. The pipe channel performs delivery of messages using per-user, site-supplied programs.

To facilitate program delivery, you must first register programs as able to be invoked by the pipe channel. Do this by using the `imsimta` program utility. This utility gives a unique name to each command that you register as able to be invoked by the pipe channel. End users can then specify the method name as a value of their `mailprogramdeliveryinfo` LDAP attribute.

For example, to add a UNIX command `myprocmail` as a program that can be invoked by a user, you would first register the command by using the `imsimta` program utility as shown in the following example. This example registers a program called `myprocmail` that executes the program `procmail` with the arguments `-d username` and executes as the user:

```
imsimta program -a -m myprocmail -p procmail -g "-d %s" -e user
```

Make sure the executable exists in the `programs` directory `msg-svr-base/data/site-programs`. Make sure also that the execute permissions are set to “others.”

To enable a user to access the program, the user’s LDAP entry must contain the following attributes and values:

```
maildeliveryoption: program  
mailprogramdeliveryinfo: myprocmail
```

For more information about the `imsimta` program utility, see the [Sun Java System Messaging Server 6.3 Administration Reference](#).

Alternative delivery programs must conform to the following exit code and command-line argument restrictions:

Exit Code Restrictions. Delivery programs invoked by the pipe channel must return meaningful error codes so that the channel knows whether to dequeue the message, deliver for later processing, or return the message.

If the subprocess exits with an exit code of 0 (`EX_OK`), the message is presumed to have been delivered successfully and is removed from the MTA queues. If it exits with an exit code of 71, 74, 75, or 79 (`EX_OSERR`, `EX_IOERR`, `EX_TEMPFAIL`, or `EX_DB`), a temporary error is presumed to have occurred and delivery of the message is deferred. If any other exit code is returned, then the message will be returned to its originator as undeliverable. These exit codes are defined in the system header file `sysexits.h`.

Command Line Arguments. Delivery programs can have any number of fixed arguments as well as the variable argument, %s, representing the user name for programs executed by the user or username+domain for programs executed by the postmaster, “inetmail.” For example, the following command line delivers a recipient’s mail using the program procmail:

```
/usr/lib/procmail -d %s
```

13.3 To Configure the Native (/var/mail) Channel

An option file may be used to control various characteristics of the native channel. This native channel option file must be stored in the MTA configuration directory and named `native_option` (for example, `msg-svr-base/config/native_option`).

Option files consist of several lines. Each line contains the setting for one option. An option setting has the form:

option=value

The *value* may be either a string or an integer, depending on the option’s requirements.

TABLE 13-2 Local Channel Options

Options	Descriptions
FORCE_CONTENT_LENGTH (0 or 1; UNIX only)	If FORCE_CONTENT_LENGTH=1, then the MTA adds a Content-length: header line to messages delivered to the native channel, and causes the channel not to use the “>From” syntax when “From” is at the beginning of the line. This makes local UNIX mail compatible with Sun’s newer mail tools, but potentially incompatible with other UNIX mail tools.
FORWARD_FORMAT (string)	Specifies the location of the users’ .forward files. The string %u indicates that it is substituted in each user id. The string %h indicates that it is substituted in each user’s home directory. The default behavior, if this option is not explicitly specified, corresponds to: FORWARD_FORMAT=%h/.forward
REPEAT_COUNT (integer) SLEEP_TIME (integer)	In case the user’s new mail file is locked by another process when the MTA tries to deliver the new mail, these options provide a way to control the number and frequency of retries the native channel program should attempt. If the file can not be opened after the number of retries specified, the messages remain in the native queue and the next run of the native channel attempts to deliver the new messages again. The REPEAT_COUNT option controls how many times the channel programs attempt to open the mail file before giving up. REPEAT_COUNT defaults to 30, (30 attempts). The SLEEP_TIME option controls how many seconds the channel program waits between attempts. SLEEP_TIME defaults to 2 (two seconds between retries).

TABLE 13-2 Local Channel Options *(Continued)*

Options	Descriptions
SHELL_TIMEOUT (integer)	Controls the length of time in seconds the channel waits for a user's shell command in a .forward to complete. Upon such time-outs, the message are returned to the original sender with an error message resembling "Time-out waiting for <i>user</i> 's shell command <i>command</i> to complete." The default is 600 (10 minutes).
SHELL_TMPDIR (directory-specific)	Controls the location where the local channel creates its temporary files when delivering to a shell command. By default, such temporary files are created in users' home directories. Using this option, the administrator may instead choose the temporary files to be created in another (single) directory. For example: SHELL_TMPDIR=/tmp

13.4 To Temporarily Hold Messages Using the Hold Channel

The hold channel is used to hold the messages of a recipient temporarily prevented from receiving new messages. Messages may be held because a user's name is being changed or their mailbox is being moved from one mailhost or domain to another. There may also be other reasons to temporarily hold messages.

When messages are to be held, they are directed to the hold channel, in the *msg-svr-base/queue/hold* directory, using the same mechanism used to direct messages to the reprocess channel. In this way, the envelope To: addresses are unchanged. The messages are written to the hold channel queue, in the *msg-server/queue/hold* directory, as *ZZxxx.HELD* files. This prevents them from being seen by the job controller, and thus they are "held." Use the *imsimta qm dir -held* command to view a list of .HELD files. These messages can be selected and released using the *imsimta qm release* command. Releasing them changes their name to *ZZxxx.00* and informs the job controller. The messages are then processed by the master program associated with the hold channel, *reprocess.exe*. Thus the message (and the To: addresses) are processed using the normal rewriting machinery.

For more information on the *imsimta qm* command, see "[imsimta qm](#)" in *Sun Java System Messaging Server 6.3 Administration Reference*.

13.5 The Conversion Channel

The conversion channel allows you to perform arbitrary body part-by-body part processing on specified messages flowing through the MTA. (Note that a body part is different than a message in that a message can contain multiple body parts as, for instance, in an attachment. Also, body parts are specified and delineated by MIME headers.) This processing can be done by any site-supplied programs or command procedures and may do such things such as convert text or images from one format to another, virus scanning, language translation and so forth. Various message types of the MTA traffic are selected for conversion, and specific processes and programs can be specified for each type of message body part.

The prerequisite for using this chapter is understanding the concept of channels (see “[8.5 Channels](#)” on page 197). For supplemental information on virus scanning using the conversion channel, refer to *Virus Screening with the iPlanet Messaging Server Conversion Channel*.

Implementing the conversion channel consists of A) selecting message traffic for processing, and B) specifying how different messages will be processed. These procedures will be discussed in further detail.

Note – A default conversion channel is automatically created in the MTA configuration file (`imta.cnf`). This channel can be used as is and requires no modification.

This section consists of the following sections:

- “[13.5.1 MIME Overview](#)” on page 432
- “[13.5.2 Selecting Traffic for Conversion Processing](#)” on page 434
- “[13.5.3 To Control Conversion Processing](#)” on page 435
- “[13.5.4 To Bounce, Delete, Hold, Retry Messages Using the Conversion Channel Output](#)” on page 444
- “[13.5.5 Conversion Channel Example](#)” on page 446
- “[13.5.6 Automatic Arabic Character Set Detection](#)” on page 450

13.5.1

MIME Overview

The conversion channel makes extensive use of the MIME (Multipurpose Internet Mail Extensions) header lines. Knowledge of message construction and MIME header fields is required. For complete information on MIME, refer to [RFCs 1806, 2045 through 2049, and 2183](#). (<http://www.faqs.org/rfcs/>) A short overview of MIME is presented here for convenience.

13.5.1.1

Message Construction

A simple message consists of a header and a body. The header is at the top of the message and contains certain control information such as date, subject, sender, and recipient. The body is everything after the first blank line after the header. MIME specifies a way to construct more complex messages which can contain multiple body parts, and even body parts nested within body parts. Messages like these are called multi-part messages, and, as mentioned earlier, the conversion channel performs body part-by-body part processing of messages.

13.5.1.2

MIME Headers

The MIME specification defines a set of header lines for body parts. These include `MIME-Version`, `Content-type`, `Content-Transfer-Encoding`, `Content-ID`, and

Content-disposition. The conversion channel uses the Content-type and Content-disposition headers most frequently. An example of some MIME header lines is shown below:

```
Content-type: APPLICATION/wordperfect5.1;name=Poem.wpc  
Content-transfer-encoding: BASE64  
Content-disposition: attachment; filename=Poem.wpc  
Content-description: "Project documentation Draft1 wordperfect format"
```

Note – MIME header lines are not the same as general, non-MIME header lines such as To:, Subject: and From:. Basically, for Conversion channel discussion, MIME header lines start with the string Content-.

Content-type Header

The MIME Content-Type header describes the content of the body-part. The Content-Type header format (with an example) is shown below:

Content-type: *type/subtype; parameter=value; parameter=value...*

type describes the type of content of the body part. Examples of type are Text, Multipart, Message, Application, Image, Audio, and Video.

subtype further describes content type. Each Content-type has its own set of subtypes. For examples: text/plain, application/octet-stream, and image/jpeg. Content Subtypes for MIME mail are assigned and listed by the IANA (Internet Assigned Numbers Authority). A copy of the list is at <http://www.iana.org/assignments/media-types>.

parameter is specific to Content-type/subtype pairs. For example, the charset and the name parameters are shown below:

```
Content-type: text/plain; charset=us-ascii  
Content-type: application/msword; name=temp.doc
```

The charset parameter specifies a character set for a textual message. The name parameter gives a suggested file name to be used if the data were to be written to a file.

Note – Content-Type values, subtypes, and parameter names are case-insensitive.

Content-disposition Header

The MIME Content-disposition header provides presentation information for the body-part. It is often added to attachments specifying whether the attachment body part should be displayed (inline) or presented as a file name to be copied (attachment). The Content-disposition header has the following format:

`Content-disposition: disposition_type; parameter=value;parameter=value...`

disposition_type is usually `inline` (display the body part) or `attachment` (present as file to save.) `Attachment` usually has the parameter `filename` with a value specifying the suggested name for the saved file.

For details on the `Content-disposition` header, refer to RFC2183.

13.5.2 Selecting Traffic for Conversion Processing

Unlike other MTA channels, the conversion channel is not normally specified in an address or MTA rewrite rule. Instead, messages are sent through the conversion channel if they meet the criteria specified in the `CONVERSIONS` mapping table (the name of the `mappings` file is specified by the parameter `IMTA_MAPPING_FILE` in the `imta_tailor` file. The default is `msg_srv_base/conversions`). Entries to the table have the following format:

`IN-CHAN=source-channel;OUT-CHAN=destination-channel;CONVERT Yes/No`

As the MTA processes each message it probes the `CONVERSIONS` mapping table (if one is present). If the *source-channel* is the channel from which the message is coming and *destination-channel* is the channel to which the message is going, then the action following `CONVERT` is taken. `Yes` means the MTA diverts the message through the conversion channel on its way to its *destination-channel*. If no match is found, or if `No` was specified, the message will be queued to the regular destination channel.

If you route messages to the conversion channel using conversion mappings your other mappings should continue to work. However, if you use rewrite rules to route messages to the conversion channel, you may have to adjust your mappings to accommodate what you've done.

Note – An address of the form `user@conversion.hostname` or `user@conversion` will be routed through the conversion channel, regardless of the `CONVERSIONS` mapping table.

The following example routes all non-internal messages—messages originating from, or destined to, the Internet—through the conversion channel.

CONVERSIONS

```
IN-CHAN=tcp_local;OUT-CHAN=*;CONVERT    Yes
IN-CHAN=*;OUT-CHAN=tcp_local;CONVERT    Yes
```

The first line specifies that messages coming from the `tcp_local` channel will be processed. The second line specifies that messages going to the `tcp_local` channel will also be processed. The `tcp_local` channel handles all messages going to and coming from the Internet. Since the default is to not go through the conversion channel, any other messages won't go through the conversion channel.

Note that this is a very basic table, and that it might not be sufficient for a site with a more customized configuration, for example, one using multiple outbound-to-the-Internet `tcp_*` channels, or using multiple inbound-from-the-Internet `tcp_*` channels.

13.5.3 To Control Conversion Processing

This section describes how to control conversion processing. It consists of the following subsections:

- “13.5.3.1 Conversion Channel Information Flow” on page 437
- “13.5.3.2 To Use Conversion Channel Environmental Variables” on page 437
- “13.5.3.3 To Use Conversion Channel Output Options” on page 441
- “13.5.3.4 Headers in an Enclosing MESSAGE/RFC822 Part” on page 443
- “13.5.3.5 To Call Out to a Mapping Table from a Conversion Entry” on page 443

When a message is sent to the conversion channel, it is processed body part-by-body part. Processing is controlled by the MTA conversions file, which is specified by the `IMTA_CONVERSION_FILE` option in the `imta_tailor` file (default: `msg-svr-base/conversions`). The conversions file consists of line-separated entries that 1) qualify which types of body parts will be processed, and 2) how they will be processed.

Each entry consists of one or more lines containing one or more `name=value` parameter clauses. Where the name in the `>` parameter clauses is one of the parameters in Table 11-5. The values in the parameter clauses conform to MIME conventions. Every line except the last must end with a semicolon (`;`). A physical line in this file is limited to 252 characters. You can split a logical line into multiple physical lines using the back slash (`\`) continuation character. Entries are terminated either by a line that does not end in a semicolon, one or more blank lines, or both.

Below is a simple example of a conversion file entry:

EXAMPLE 13-1 conversions File Entry

```
out-chan=ims-ms; in-type=application; in-subtype=wordperfect5.1;
out-type=application; out-subtype=msword; out-mode=block;
command="/usr/bin/convert -in=wordp -out=msword 'INPUT_FILE' 'OUTPUT_FILE'"
```

The clauses `out-chan=ims-ms`; `in-type=application`; `in-subtype=wordperfect5.1` qualify the body part. That is, they specify the type of part to be converted. The header of each part is read and its `Content-Type:` and other header information is extracted. The entries in the conversion file are then scanned in order from first to last; any `in-*` parameters present, and the `OUT-CHAN` parameter, if present, are checked. If all of these parameters match the corresponding information for the body part being processed, then the conversion specified by the `command=` or `delete=` clause is performed, and the `out-*` parameters are set.

If no match occurs, then the part is matched against the next conversions file entry. Once all body parts have been scanned and processed (assuming there is a qualifying match), then the message is sent onwards to the next channel. If there are no matches, no processing occurs, and the message is sent to the next channel.

`out-chan=ims-ms` specifies that only message parts destined for the `ims-ms` channel will be converted. `in-type=application` and `in-subtype=wordperfect5.1` specifies that the MIME Content-type header for the message part must be `application/wordperfect5.1`.

Message parts can be further qualified with additional `in-*` parameters. (See [Table 13–6](#).) The entry above will trigger conversion actions on a message part which has the following MIME header lines:

```
Content-type: APPLICATION/wordperfect5.1;name=Draft1.wpc
Content-transfer-encoding: BASE64
Content-disposition: attachment; filename=Draft1.wpc
Content-description: "Project documentation Draft1 wordperfect format"
```

After the three conversion file qualifying parameters in [Example 13–1](#), the next two parameters, `out-type=application` and `out-subtype=msword`, specify replacement MIME header lines to be attached to the “processed” body part. `out-type=application` and `out-subtype=msword` specify that the MIME Content-type/subtype of the outgoing message be `application/msword`.

Note that since the `in-type` and `out-type` parameters are the same, `out-type=application` is not necessary since the conversion channel defaults to the original MIME labels for outgoing body parts. Additional MIME labels for outgoing body parts can be specified with additional output parameters.

`out-mode=block` ([Example 13–1](#)) specifies the file type that the site-supplied program will return. In other words, it specifies how the file will be stored and how the conversion channel should be read back in the returned file. For example, an `html` file is stored in text mode, while an `.exe` program or a `zip` file is stored in block/binary mode. Mode is a way of describing that the file being read is in a certain storage format.

The final parameter in [Example 13–1](#) specifies the action to take on the body part:

```
command="/usr/bin/convert -in=wordp -out=msword 'INPUT_FILE' 'OUTPUT_FILE'"
```

The `command=` parameter specifies that a program will execute on the body part. `/usr/bin/convert` is the hypothetical command name; `-in=wordp` and `-out=msword` are hypothetical command line arguments specifying the format of the input text and output text; `INPUT_FILE` and `OUTPUT_FILE` are conversion channel environmental variables (see [“13.5.3.2 To Use Conversion Channel Environmental Variables” on page 437](#)) program should store its converted body part.

Note – Envelope originator and recipient information is now provided as x-envelope-from and x-envelope-to fields respectively when a file containing the outer message header is requested by a regular conversion entry.

Instead of executing a command on the body part, the message part can simply be deleted by substituting `DELETE=1` in place of the `command` parameter.

Note – Whenever the conversions file is modified, you must recompile the configuration (see “[10.1 Compiling the MTA Configuration](#)” on page 233).).

13.5.3.1

Conversion Channel Information Flow

The flow of information is as follows: a message containing body parts comes into the conversion channel. The conversion channel parses the message, and processes the parts one by one. The conversion channel then qualifies the body part, that is, it determines if it should be processed or not by comparing its MIME header lines to the *qualifying parameters* ([Table 13–6](#)). If the body part qualifies, the conversion processing commences.

If MIME or body part information is to be passed to the conversion script, it is stored in an environmental variable (“[13.5.3.2 To Use Conversion Channel Environmental Variables](#)” on page 437) as specified by *information passing parameters* ([Table 13–6](#)).

At this point, an action specified by an *action parameter*, ([Table 13–6](#)) is taken on the body part. Typically the action is that the body part be deleted or that it be passed to a program wrapped in a script. The script processes the body part and then sends it back to the conversion channel for reassembling into the post-processed message. The script can also send information to the conversion channel by using the conversion channel *output options* ([Table 13–4](#)). This can be information such as new MIME header lines to add to the output body part, error text to be returned to the message sender, or special directives instructing the MTA to initiate some action such as bounce, delete, or hold a message.

Finally, the conversion channel replaces the header lines for the output body part as specified by the *output parameters* ([Table 13–6](#)).

13.5.3.2

To Use Conversion Channel Environmental Variables

When operating on message body parts, it is often useful to pass MIME header line information, or entire body parts, to and from the site-supplied program. For example, a program may require `Content-type` and `Content-disposition` header line information as well as a message body part. Typically a site-supplied program’s main input is a message body part which is read from a file. After processing the body part, the program will need to write it to a file from which the conversion channel can read it. This type of information passing is done by using conversion channel environmental variables.

Environmental variables can be created in the conversions file using the `parameter-symbol-*` parameter or by using a set of pre-defined conversion channel environmental variables (see “[13.5.3.3 To Use Conversion Channel Output Options](#)” on page 441).

The following conversions file entry and incoming header show how to pass MIME information to the site-supplied program using environment variables.

conversions file entry:

```
in-channel=*; in-type=application; in-subtype=*
parameter-symbol-0=NAME; parameter-copy-0=*
dparameter-symbol-0=FILENAME; dparameter-copy-0=*
message-header-file=2; original-header-file=1;
override-header-file=1; override-option-file=1;
command="/bin/viro-scan500.sh "INPUT_FILE" "OUTPUT_FILE"
```

Incoming header:

```
Content-type: APPLICATION/msword; name=Draft1.doc
Content-transfer-encoding: BASE64
Content-disposition: attachment; filename=Draft1.doc
Content-description: "Project documentation Draft1 msword format"
```

`in-channel=*`; `in-type=application`; `in-subtype=*` specify that a message body part from any input channel of type `application` will be processed.

`parameter-symbol-0=NAME` specifies that value of the `Content-type` parameter name, if present (`Draft1.doc` in our example), be stored in an environment variable called `NAME`.

`parameter-copy-0=*` specifies that all `Content-type` parameters of the input body part be copied to the output body part.

`dparameter-symbol-0=FILENAME` specifies that the value of the `Content-disposition` parameter `filename` (`Draft1.doc` in our example), be stored in an environment variable called `FILENAME`.

`dparameter-copy-0=*` specifies that all `Content-disposition` parameters of the input body part be copied to the output body part.

`message-header-file=2` specifies that the original header of the message as a whole (the outermost message header) be written to the file specified by the environment variable `MESSAGE_HEADERS`.

`original-header-file=1` specifies that the original header of the enclosing `MESSAGE/RFC822` part are written to the file specified by the environment variable `INPUT_HEADERS`.

`override-header-file=1` specifies that MIME headers are read from the file specified by environmental variable `OUTPUT_HEADERS`, overriding the original MIME header lines in the enclosing MIME part. `$OUTPUT_HEADERS` is an on-the-fly temporary file created at the time conversion runs. A site-supplied program would use this file to store MIME header lines changed during the conversion process. The conversion channel would then read the MIME header lines from this file when it re-assembles the body part. Note that only MIME header lines can be modified. Other general, non-MIME header lines cannot be altered by the conversion channel.

`override-option-file=1` specifies that the conversion channel read *conversion channel options* from the file named by the `OUTPUT_OPTIONS` environmental variable. See “[13.5.3.3 To Use Conversion Channel Output Options](#)” on page 441.

`command="msg-svr-base/bin/viro-scan500.sh"` specifies the command to execute on the message body part.

TABLE 13-3 Conversion Channel Environment Variables

Environment Variable	Description
<code>ATTACHMENT_NUMBER</code>	Attachment number for the current part. This has the same format as the <code>ATTACHMENT-NUMBER</code> conversion match parameter.
<code>CONVERSION_TAG</code>	The current list of active conversion tags. This corresponds to the <code>TAG</code> conversion match parameter.
<code>INPUT_CHANNEL</code>	The channel that enqueued the message to the conversion channel. This corresponds to the <code>IN-CHANNEL</code> conversion match parameter.
<code>INPUT_ENCODING</code>	Encoding originally present on the body part.
<code>INPUT_FILE</code>	Name of the file containing the original body part. The site-supplied program should read this file.
<code>INPUT_HEADERS</code>	Name of the file containing the original header lines for the body part. The site-supplied program should read this file.
<code>INPUT_TYPE</code>	MIME <code>Content-type</code> of the input message part.
<code>INPUT_SUBTYPE</code>	MIME content subtype of the input message part.
<code>INPUT_DESCRIPTION</code>	MIME <code>content-description</code> of the input message part.
<code>INPUT_DISPOSITION</code>	MIME <code>content-disposition</code> of the input message part.
<code>MESSAGE_HEADERS</code>	Name of the file containing the original outermost header for an enclosing message (not just the body part) or the header for the part's most immediately enclosing MESSAGE/RFC822 part. The site-supplied program should read this file.

TABLE 13-3 Conversion Channel Environment Variables *(Continued)*

Environment Variable	Description
OUTPUT_CHANNEL	The channel the message is headed for. This corresponds to the OUT-CHANNEL conversion match parameter.
OUTPUT_FILE	Name of the file where the site-supplied program should store its output. The site-supplied program should create and write this file.
OUTPUT_HEADERS	Name of the file where the site-supplied program should store MIME header lines for an enclosing part. The site-supplied program should create and write this file. Note that file should contain actual MIME header lines (not option=value lines) followed by a blank line as its final line. Note also that only MIME header lines can be modified. Other general, non-MIME header lines cannot be altered by the conversion channel.
OUTPUT_OPTIONS	Name of the file from which the site-supplied program should read conversion channel options. See “ 13.5.3 To Use Conversion Channel Output Options ” on page 441.
PART_NUMBER	The part number for the current part. This has the same format as the PART-NUMBER conversion match parameter.
PART_SIZE	The size in bytes of the part being processed.

Mail Conversion Tags

Mail conversion tags are special tags which are associated with a particular recipient or sender. When a message is being delivered, the tag is visible to the conversion channel program, which may make use of it for special processing. Conversion tags are stored in the LDAP directory.

Mail conversion tags could be used as follows: the administrator can set up selected users with a mail conversion tag value of `harmonica`. The administrator then has a conversion channel setup which, when processing that mail, will detect the presence of the tag and the value of `harmonica`. When that happens, the program will perform some arbitrary function.

Mail conversion tags can be set on a per user or a per domain basis. The recipient LDAP attribute at the domain level is `MailDomainConversionTag` (modifiable with the MTA option `LDAP_DOMAIN_ATTR_CONVERSION_TAG`). At the user level it is `MailConversionTag` (modifiable with the MTA option `LDAP_CONVERSION_TAG`). Both of these attributes can be multivalued with each value specifying a different tag. The set of tags associated with a given recipient is cumulative, that is, tags set at the domain level are combined with tags set at the user level.

Sender-based conversion tags can be set with the MTA options `LDAP_SOURCE_CONVERSION_TAG` and `LDAP_DOMAIN_ATTR_SOURCE_CONVERSION_TAG`, which specify user and domain level LDAP attributes respectively for conversion tags associated with these source address. There is no default attribute for either of these options.

Two new actions are available to system Sieves: `addconversiontag` and `setconversiontag`. Both accept a single argument: A string or list of conversion tags. `addconversiontag` adds the conversion tag(s) to the current list of tags while `setconversiontag` empties the existing list before adding the new ones. Note that these actions are performed very late in the game so `setconversiontag` can be used to undo all other conversion tag setting mechanisms. These allow you put conversion tags in the Sieves filters.

The Sieve envelope test accepts `conversiontag` as an envelope field specifier value. The test checks the current list of tags, one at a time. Note that the `:count` modifier, if specified, allows checking of the number of active conversion tags. This type of envelope test is restricted to system Sieves. Also note that this test only sees the set of tags that were present prior to Sieve processing—the effects of `setconversiontag` and `addconversiontag` actions are not visible.

Including Conversion Tag Information in Various Mapping Probes

A new MTA option, `INCLUDE_CONVERSIONTAG`, has been added to selectively enable the inclusion of conversion tag information in various mapping probes. This is a bit-encoded value. The bits are assigned are shown in the table below. In all cases the current set of tags appears in the probe as a comma separated list.

Position	Value	Mapping
0	1	<code>CHARSET_CONVERSION</code> - added as ;TAG= field before ; <code>CONVERT</code> .
1	2	<code>CONVERSION</code> - added as ;TAG= field before ; <code>CONVERT</code>
2	4	<code>FORWARD</code> - added just before current address (delim)
3	8	<code>ORIG_SEND_ACCESS</code> - added at end of probe (delim)
4	16	<code>SEND_ACCESS</code> - added at end of probe (delim)
5	32	<code>ORIG_MAIL_ACCESS</code> - added at end of probe (delim)
6	64	<code>MAIL_ACCESS</code> - added at end of probe (delim)

13.5.3.3

To Use Conversion Channel Output Options

Conversion channel output options (Table 13-4) are dynamic variables used to pass information and special directives from the conversion script to the conversion channel. For example, during body part processing the script may want to send a special directive asking the conversion channel to bounce the message and to add some error text to the returned message stating that the message contained a virus.

The output options are initiated by setting `OVERRIDE-OPTION-FILE=1` in the desired conversion entry. Output options are then set by the script as needed and stored in the environmental variable file, `OUTPUT_OPTIONS`. When the script is finished processing the body part, the conversion channel reads the options from the `OUTPUT_OPTIONS` file.

The `OUTPUT_OPTION` variable is the name of the file from which the conversion channel reads options. Typically it is used as an on-the-fly temporary file to pass information. The example below shows a script that uses output options to return an error message to a sender who mailed a virus.

```
/usr/local/bin/viro_screen2k $INPUT_FILE # run the virus screener

if [ $? -eq 1 ]; then
    echo "OUTPUT_DIAGNOSTIC='Virus found and deleted.'" > $OUTPUT_OPTIONS
    echo "STATUS=178029946" >> $OUTPUT_OPTIONS
else
    cp $INPUT_FILE $OUTPUT_FILE # Message part is OK
fi
```

In this example, the system diagnostic message and status code are added to the file defined by `$OUTPUT_OPTIONS`. If you read the `$OUTPUT_OPTIONS` temporary file out you would see something like:

```
OUTPUT_DIAGNOSTIC="Virus found and deleted."
STATUS=178029946
```

The line `OUTPUT_DIAGNOSTIC='Virus found and deleted'` tells the conversion channel to add the text `Virus found and deleted` to the message.

`178029946` is the `PMDF__FORCERETURN` status per the `pmdf_err.h` file which is found in the `msg-svr-base/include/deprecated/pmdf_err.h`. This status code directs the conversion channel to bounce the message back to the sender. (For more information on using special directives refer to “[13.5.4 To Bounce, Delete, Hold, Retry Messages Using the Conversion Channel Output](#)” on page 444)

A complete list of the output options is shown below.

TABLE 13-4 Conversion Channel Output Options

Option	Description
<code>OUTPUT_TYPE</code>	MIME content type of the output message part.
<code>OUTPUT_SUBTYPE</code>	MIME content subtype of the output message part.
<code>OUTPUT_DESCRIPTION</code>	MIME content description of the output message part.
<code>OUTPUT_DIAGNOSTIC</code>	Text to include as part of the message sent to the sender if a message is forcibly bounced by the conversion channel.
<code>OUTPUT_DISPOSITION</code>	MIME content-disposition of the output message part.

TABLE 13-4 Conversion Channel Output Options *(Continued)*

Option	Description
OUTPUT_ENCODING	MIME content transfer encoding to use on the output message part.
OUTPUT_MODE	MIME Mode with which the conversion channel should write the output message part, hence the mode with which recipients should read the output message part.
STATUS	Exit status for the converter. This is typically a special directive initiating some action by the conversion channel. A complete list of directives can be viewed in <i>msg-srv-base/include/deprecated/pmdf_err.h</i>

13.5.3.4 Headers in an Enclosing MESSAGE/RFC822 Part

When performing conversions on a message part, the conversion channel has access to the header in an enclosing MESSAGE/RFC822 part, or to the message header if there is no enclosing MESSAGE/RFC822 part. Information in the header may be useful for the site-supplied program.

If an entry is selected that has ORIGINAL-HEADER-FILE=1, then all the original header lines of the enclosing MESSAGE/RFC822 part are written to the file represented by the ORIGINAL_HEADERS environment variable. If OVERRIDE-HEADER-FILE=1, then the conversion channel will read and use as the header on that enclosing part the contents of the file represented by the ORIGINAL_HEADERS environment variable.

13.5.3.5 To Call Out to a Mapping Table from a Conversion Entry

out-parameter-* values may be stored and retrieved in an arbitrarily named mapping table. This feature is useful for renaming attachments sent by clients that send all attachments with a generic name like att.dat regardless of whether they are postscript, msword, text or whatever. This is a generic way to relabel the part so that other clients (Outlook for example) will be able to open the part by reading the extension.

The syntax for retrieving a parameter value from a mapping table is as follows:

"mapping-table-name:mapping-input[\$Y,\$N]"

\$Y returns a parameter value. If there is no match found or the match returns \$N, then that parameter in the conversions file entry is ignored or treated as a blank string. Lack of a match or a \$N does not cause the conversion entry itself to be aborted.

Consider the following mapping table:

X-ATT-NAMES

postscript	temp.PS\$Y
wordperfect5.1	temp.WPC\$Y
msword	temp.DOC\$Y

The following conversion entry for the above mapping table results in substituting generic file names in place of specific file names on attachments:

```
out-chan=tcp_local; in-type=application; in-subtype=*
in-parameter-name-0=name; in-parameter-value-0=*
out-type=application; out-subtype='INPUT-SUBTYPE';
out-parameter-name-0=name;
out-parameter-value-0="X-ATT-NAMES:\\"INPUT_SUBTYPE\\\"";
command="cp \"INPUT_FILE\" \"OUTPUT_FILE\""
```

In the example above, `out-chan=tcp_local; in-type=application; in-subtype=*` specifies that a message to be processed must come from the `tcp_local` channel with the `content-type` header of `application/*` (* specifies that any subtype would do).

`in-parameter-name-0=name; in-parameter-value-0=*` additionally specifies that the message must have a `content-type` parameter called `name=*` and that any value for that parameter will be accepted (again, * specifies that any parameter value would do.)

`out-type=application;` specifies that the MIME `Content-type` parameter for the post-processing message be `application`.

`out-subtype='INPUT-SUBTYPE'`; specifies that the MIME `subtype` parameter for the post-processing body part be the `INPUT-SUBTYPE` environmental variable, which is the original value of the input subtype. Thus, if you wanted change

`Content-type: application/xxxx; name=foo.doc`

to

`Content-type: application/msword; name=foo.doc`

then you would use

`out-type=application; out-subtype=msword`

`out-parameter-name-0=name;` specifies that the output body part will have a MIME `Content-type` `name=` parameter.

`out-parameter-value-0='X-ATT-NAMES:\\"INPUT_SUBTYPE\\\"';` says to take the value of the `INPUT_SUBTYPE` variable (that is, the original `content-type` header subtype value of the original body part) and search the mapping table `X-ATT-NAMES`. If a match is found, the `content-type` parameter specified by `out-parameter-name-0` (that is, `name`) receives the new value specified in the `X-ATT-NAMES` mapping table. Thus, if the original subtype was `msword`, the value of the `name` parameter will be `temp.DOC`.

13.5.4 To Bounce, Delete, Hold, Retry Messages Using the Conversion Channel Output

This section describes how to use the conversion channel options to bounce, delete, or hold messages. The basic procedure is as follows:

1. Set OVERRIDE-OPTION-FILE=1 in the appropriate conversions file entry. This tells the conversion channel to read the output options from the OUTPUT_OPTIONS file.
2. Use the conversion script to determine what action is required on a particular message body part.
3. In the script, specify the special directive for that action by writing the STATUS=*directive_code* option in the OUTPUT_OPTIONS file.

A complete listing of special directives can be found in *msg-srv-base/include/deprecated/pmdf_err.h*. The ones commonly used by the conversion channel are:

TABLE 13-5 Special Directives Commonly Used By the Conversion Channel

NAME	Hex Value	Decimal Value
PMDF__FORCEHOLD	0x0A9C86AA	178030250
PMDF__FORCERETURN	0x0A9C857A	178029946
PMDF__FORCEDELETE	0x0A9C8662	178030178
PMDF__FORCEDISCARD	0x0A9C86B3	178030259
PMDF__AGN	0x0A9C809A	178028698

We will explain the functions of these directives using examples.

13.5.4.1 To Bounce Messages

To bounce a message using the conversion channel set OVERRIDE-OPTION-FILE=1 in the appropriate conversions file entry and add the following line to your conversion script:

```
echo "STATUS=178029946" >> $OUTPUT_OPTIONS
```

If you wish to add a short text string to the bounced message add the following line to the conversion script:

```
echo OUTPUT_DIAGNOSTIC=text-string >> $OUTPUT_OPTIONS
```

where text string is something like: “The message sent from your machine contained a virus which has been removed. Be careful about executing email attachments.”

13.5.4.2 To Conditionally Delete a Message or Its Parts

It may be useful to delete parts conditionally, depending on what they contain. This can be done using the output options. By contrast, the DELETE=1 conversion parameter clause unconditionally deletes a message part.

To delete a message part using the output options, set OVERRIDE-OPTION-FILE=1 in the appropriate conversions file entry and add the following line to your conversion script:

```
echo "STATUS=178030178" >> $OUTPUT_OPTIONS
```

Similarly, to delete the entire message you could use:

```
echo "STATUS=178030259" >> $OUTPUT_OPTIONS
```

13.5.4.3

To Hold a Message

It may be useful to hold messages conditionally, depending on what they contain. To delete a message part using the output options, set OVERRIDE-OPTION-FILE=1 in the appropriate conversions file entry and add the following line to your conversion script:

```
echo "STATUS=178030250" >> $OUTPUT_OPTIONS
```

This requests that the conversion channel hold the message as a .HELD file in the conversion channel queue.

13.5.4.4

To Cause Messages to Be Reprocessed

When a converter script encounters a temporary resource problem (for example, the system can't connect to an external server, a needed file is locked, and so on), you can use PMDF_AGN to tell the conversion channel to consider processing messages that have encountered a temporary error. The MTA will record a "Q" status message in the mail.log_current, retain the message in the conversion channel, and retry the processing later.

Add the following line to your conversion script:

```
echo "STATUS=178028698" >> $OUTPUT_OPTIONS
```

13.5.5

Conversion Channel Example

The CONVERSIONS mapping and set of conversion rules seen in examples below cause GIF, JPEG, and BITMAP files sent to the hypothetical channel `tcp_docuprint` to be converted into PostScript automatically. Several of these conversions use the hypothetical `/usr/bin/ps-converter.sh` to make that transformation. An additional rule that converts WordPerfect 5.1 files into Microsoft Word files is included.

CONVERSIONS

```
IN-CHAN=* ;OUT-CHAN=tcp_docuprint;CONVERT      Yes  
  
out-chan=ims-ms; in-type=application; in-subtype=wordperfect5.1;  
          out-type=application; out-subtype=msword; out-mode=block;  
          command="/bin/doc-convert -in=wp -out=msw   'INPUT_FILE'  'OUTPUT_FILE'"
```

```

out-chan=tcp_docuprint; in-type=image; in-subtype=gif;
out-type=application; out-subtype=postscript; out-mode=text;
command="/bin/ps-convert -in=gif -out=ps 'INPUT_FILE' 'OUTPUT_FILE'"

out-chan=tcp_docuprint; in-type=image; in-subtype=jpeg;
out-type=application; out-subtype=postscript; out-mode=text;
command="/bin/ps-convert -in=jpeg -out=ps 'INPUT_FILE' 'OUTPUT_FILE'"

out-chan=tcp_docuprint; in-type=image; in-subtype=bitmap;
out-type=application; out-subtype=postscript; out-mode=text;
command="/bin/ps-convert -in=bmp -out=ps 'INPUT_FILE' 'OUTPUT_FILE'"

```

The conversion parameters are shown below:

TABLE 13-6 Conversion Parameters

Parameter	Description
Part 1: Qualifying Parameters (Specifies the parameters for which the message must match before it will be converted.)	
OUT-CHAN, OUT-CHANNEL	Output channel to match for conversion (wildcards allowed). The conversion specified by this entry is performed only if the message is destined for this specified channel.
IN-CHAN, IN-CHANNEL	Input channel to match for conversion (wildcards allowed). The conversion specified by this entry is only performed if the message is coming from the specified channel.
IN-TYPE	Input MIME type to match for conversion (wildcards allowed). The conversion specified is performed only if this field matches the MIME type of the body part.
IN-SUBTYPE	Input MIME subtype to match for conversion (wildcards allowed). The conversion specified by this entry is performed only if this field matches the MIME subtype of the body part.
IN-PARAMETER-NAME- <i>n</i>	Specifies the name of the Input MIME Content-Type parameter that must match for conversion; The <i>n</i> = 0, 1, 2.... is used to optionally pair the specified parameter name requirement with a value required by using IN-PARAMETER-VALUE- <i>n</i> with the same value of <i>n</i> .
IN-PARAMETER-VALUE- <i>n</i>	Specifies the value required of the input MIME Content-Type parameter whose name is specified in the corresponding IN-PARAMETER-NAME- <i>n</i> . The conversion specified by this entry is performed only if the input body part has the content-type parameter specified by the corresponding IN-PARAMETER-NAME- <i>n</i> and its value matches the value of this parameter. Wildcards allowed.
IN-PARAMETER-DEFAULT- <i>n</i>	Default value to use if the input MIME Content-Type parameter specified by the corresponding IN-PARAMETER-NAME- <i>n</i> is not present.
IN-DISPOSITION	Input MIME Content-Disposition to match for conversion.

TABLE 13–6 Conversion Parameters (*Continued*)

Parameter	Description
IN-DPARAMETER-NAME- <i>n</i>	Specifies the name of the Input MIME Content-Disposition parameter that must match for conversion; The <i>n</i> = 0, 1, 2,... is used to optionally pair the specified parameter name requirement with a value required by using IN-DPARAMETER-VALUE- <i>n</i> with the same value of <i>n</i> .
IN-DPARAMETER-VALUE- <i>n</i>	Specifies the value required of the input MIME Content-Disposition parameter whose name is specified in the corresponding IN-DPARAMETER-NAME- <i>n</i> . The conversion specified by this entry is performed only if the input body part has the Content-Disposition parameter specified by the corresponding IN-DPARAMETER-NAME- <i>n</i> and its value matches the value of this parameter. Wildcards allowed.
IN-DPARAMETER-DEFAULT- <i>n</i>	Default value to use if the input MIME Content-Disposition parameter specified by the corresponding IN-DPARAMETER-NAME- <i>n</i> is not present.
IN-DESCRIPTION	Input MIME Content-Description to match for conversion.
IN-SUBJECT	Input Subject from enclosing MESSAGE/RFC822 part.
TAG	Input tag, as set by a mail list CONVERSION_TAG parameter.

Part 2: Output Parameters (Specify the body part's post-conversion output settings.)

OUT-TYPE	Output MIME type if it is different than the input type.
OUT-SUBTYPE	Output MIME subtype if it is different than the input subtype.
OUT-PARAMETER-NAME- <i>n</i>	Specifies the name of a content-type parameter which will be set on the output body part.
OUT-PARAMETER-VALUE- <i>n</i>	Output MIME Content-Type parameter value corresponding to OUT-PARAMETER-NAME- <i>n</i> .
PARAMETER-COPY- <i>n</i>	Specifies the name of a content-type parameter which should be copied from the input body part to the output body part.
OUT-DISPOSITION	Output MIME Content-Disposition if it is different than the input MIME Content-Disposition.
OUT-DPARAMETER-NAME- <i>n</i>	Output MIME Content-Disposition parameter name; <i>n</i> =0, 1, 2...
OUT-DPARAMETER-VALUE- <i>n</i>	Output MIME Content-Disposition parameter value corresponding to OUT-DPARAMETER-NAME- <i>n</i> .
DPARAMETER-COPY- <i>n</i>	A list of the Content-Disposition: parameters to copy from the input body part's Content-Disposition: parameter list to the output body part's Content-Disposition: parameter list; <i>n</i> = 0, 1, 2,... Takes as argument the name of the MIME parameter to copy, as matched by an IN-PARAMETER-NAME- <i>n</i> clause. Wildcards may be used in the argument. In particular, an argument of * means to copy all the original Content-Disposition: parameters.

TABLE 13-6 Conversion Parameters *(Continued)*

Parameter	Description
OUT-DESCRIPTION	Output MIME Content-Description if it is different than the input MIME Content-Description.
OUT-MODE	Mode in which to read and store the converted file. This should be BLOCK (binaries and executables) or TEXT.
OUT-ENCODING	Encoding to apply to the converted file when the message is reassembled.

Part 3: Action Parameters (Specify an action to take on a message part.)

COMMAND	Command to execute to perform conversion. Command to execute to perform conversion. This parameter is required; if no command is specified, the entry is ignored. Use / to specify paths, not \. Example: command="D:/tmp/mybat.bat"
DELETE	0 or 1. If this flag is set, the message part is deleted. (If this is the only part in a message, then a single empty text part is substituted.)
RELABEL	RELABEL=1 will relabel the MIME label to whatever is specified by the Output parameters. Relabel=0 does nothing. Usually relabelling is done on mislabeled parts (example: from Content-type: application/octet-stream to Content-type: application/msword) so users can “doubleclick” to open a part, rather than having to save the part to a file and open it with a program.
SERVICE-COMMAND	SERVICE-COMMAND=command will execute a site-supplied procedure that will operate on entire MIME message (MIME headers and content body part). Also, unlike other CHARSET-CONVERSION operations or conversion channel operations, the service-command are expected to do their own MIME disassembly, decoding, re-encoding, and reassembly. Note that this flag causes an entry to be ignored during conversion channel processing; SERVICE-COMMAND entries are instead performed during character set conversion processing. Use / to specify paths, not \. Example: command="D:/tmp/mybat.bat"

Part 4: Information Passing Parameters (Used to pass information to and from the site-supplied program.)

DPARAMETER-SYMBOL- <i>n</i>	Environment variable into which the Content-disposition parameter value, if present, will be stored; <i>n</i> = 0, 1, 2,... Each DPARAMETER-SYMBOL- <i>n</i> is extracted from the Content-Disposition: parameter list in order (<i>n</i> =0 is first parameter, <i>n</i> =2 second, etc.) and placed in the specified environment variable prior to executing the site-supplied program.
-----------------------------	--

TABLE 13–6 Conversion Parameters (*Continued*)

Parameter	Description
PARAMETER-SYMBOL- <i>n</i>	Specifies the name of a content-type parameter which, if present in the input body part, its value will be stored in an environment variable of the same name. If the parameter does not exist in the input body part, the environment variable will not exist in the process. For example, if you specify parameter-symbol-0=foo, and there's a content type parameter foo with value bar, you end up with an environment variable foo with value bar.
	Environment variable into which the Content-Type parameter value, if present, will be stored; <i>n</i> = 0, 1, 2... Each PARAMETER-SYMBOL- <i>n</i> is extracted from the Content-Type: parameter list in order (<i>n</i> =0 is first parameter, <i>n</i> =2 second, etc.) and placed in an environment variable of the same name prior to executing the site-supplied program. Takes as argument the variable name into which the MIME parameter to convert, as matched by an IN-PARAMETER-NAME- <i>n</i> clause.
MESSAGE-HEADER-FILE	Writes all, part, or none of the original header of a message to the file specified by the environmental variable MESSAGE_HEADERS. If set to 1, the original header of the immediately enclosing body part are written to the file specified by the environmental variable MESSAGE_HEADERS. If set to 2, the original header of the message as a whole (the outermost message header) are written to the file.
ORIGINAL-HEADER-FILE	0 or 1. If set to 1, the original header of the enclosing MESSAGE/RFC822 part (not just the body part) are written to the file represented by the environmental variable ORIGINAL_HEADERS.
OVERRIDE-HEADER-FILE	0 or 1. If set to 1, then MIME header lines are read by the conversion channel from the environmental variable OUTPUT_HEADERS, overriding the original header lines in the enclosing MIME part.
OVERRIDE-OPTION-FILE	If OVERRIDE-OPTION-FILE=1, the conversion channel reads options from the OUTPUT_OPTIONS environmental variable.
PART-NUMBER	Dotted integers: <i>a</i> . <i>b</i> . <i>c</i> ... The part number of the MIME body part.

13.5.6 Automatic Arabic Character Set Detection

A new auto_ef program was added to automatically detect Arabic character sets.

You can call the auto_ef program from the conversion channel to automatically detect and label most unlabeled or incorrectly labeled text messages in Arabic character sets. These unlabeled or mislabeled messages are usually sent from Yahoo or Hotmail in Arabic.

Without the correct character set labeling, many mail clients cannot display the messages correctly.

If a message has MIME content-type headers, the auto_ef program examines and processes only those with text/plain content type. If the message is not labeled with a MIME content-type header, then auto_ef adds a text/plain content-type unconditionally.

To activate or enable this program, you must:

▼ To Automatically Detect Arabic Character Sets

- 1 Edit your mappings file in the *msg-svr-base/config* directory to enable a conversion channel for the source and destination channel of your choosing. To enable a conversion channel for all mail coming in from the Internet to your local users, add a section to your mappings file similar to the following:

CONVERSATIONS

```
IN-CHAN=tcp*;OUT-CHAN=ims-ms;CONVERT YES
```

Note that the IN and OUT channels depend on your configuration. If you are deploying on a relay MTA, you must modify the channels to fit your configuration. For example,

```
IN-CHAN=tcp*;OUT-CHAN=tcp*;CONVERT YES
```

Or, you could turn it on for all channels as follows:

```
IN-CHAN=*;OUT-CHAN=*;CONVERT YES
```

- 2 Create a conversions file in the *msg-svr-base/config* directory that is owned and readable by the current version of Messaging Server user, and that contains the following:

```
!
in-channel=*; out-channel=.*;
in-type=text; in-subtype=.*;
parameter-copy-0=*; dparameter-copy-0=.*;
original-header-file=1; override-header-file=1;
command="msg-svr-base
/lib/arabicdetect.sh"
!
```

- 3 Compile your MTA configuration with the following command:

```
msg-svr-base/sbin/imsimta cnbuild
```

- 4 Restart with the command:

```
msg-svr-base/sbin/imsimta restart
```

13.6 Character Set Conversion and Message Reformatting

This section describes character set, formatting, and labelling conversions performed internally by the MTA. Note that some of the examples in this section use old or obsolete technology like DEC VMS, or the d channels. Although these technologies are old or obsolete, this does not make the examples DEC- or d channel-specific. The examples are still valid in describing how the conversion technology works. We will update the examples in a later release.

One very basic mapping table in Messaging Server is the character set conversion table. The name of this table is CHARSET - CONVERSION. It is used to specify what sorts of channel-to-channel character set conversions and message reformatting should be done.

On many systems there is no need to do character set conversions or message reformatting and therefore this table is not needed. Situations arise, however, where character conversions must be done. For example, sites running Japanese OpenVMS may need to convert between DEC Kanji and the ISO-2022 Kanji currently used on the Internet. Another possible use of conversions arises when multinational characters are so heavily used that the slight discrepancies between the DEC Multinational Character Set (DEC-MCS) and the ISO-8859-1 character set specified for use in MIME may become an issue, and actual conversion between the two may therefore be needed.

The CHARSET - CONVERSION mapping table can also be used to alter the format of messages. Facilities are provided to convert a number of non-MIME formats into MIME. Changes to MIME encodings and structure are also possible. These options are used when messages are being relayed to systems that only support MIME or some subset of MIME. And finally, conversion from MIME into non-MIME formats is provided in a small number of cases.

The MTA will probe the CHARSET - CONVERSION mapping table in two different ways. The first probe is used to determine whether or not the MTA should reformat the message and if so, what formatting options should be used. (If no reformatting is specified, the MTA does not bother to check for specific character set conversions.) The input string for this first probe has the general form:

IN-CHAN=*in-channel*; OUT-CHAN=*out-channel*; CONVERT

Here *in-channel* is the name of the source channel (where the message comes from) and *out-channel* is the name of the destination channel (where the message is going). If a match occurs the resulting string should be a comma-separated list of keywords. [Table 13–7](#) lists the keywords.

TABLE 13-7 CHARSET-CONVERSION Mapping Table Keywords

Keyword	Description
Always	Force conversion even the message is going to be passed through the conversion channel before going to <i>out-channel</i> .
Appledouble	Convert other MacMIME formats to Appledouble format.
Applesingle	Convert other MacMIME formats to Applesingle format.
BASE64	Switch MIME encodings to BASE64. This keyword only applies to message parts that are already encoded. Messages with Content-transfer-encoding: 7BIT or 8bit do not require any special encoding and therefore this BASE64 option will have no effect on them.
Binhex	Convert other MacMIME formats, or parts including Macintosh type and Mac creator information, to Binhex format.
Block	Extract just the data fork from MacMIME format parts.
Bottom	“Flatten” any message/rfc822 body part (forwarded message) into a message content part and a header part.
Delete	“Flatten” any message/rfc822 body part (forwarded message) into a message content part, deleting the forwarded headers.
Level	Remove redundant multipart levels from message.
Macbinary	Convert other MacMIME formats, or parts including Macintosh type and Macintosh creator information, to Macbinary format.
No	Disable conversion.
QUOTED-PRINTABLE	Switch MIME encodings to QUOTED-PRINTABLE.
Record,Text	Line wrap text/plain parts at 80 characters.
Record,Text= n	Line wrap text/plain parts at n characters.
RFC1154	Convert message to RFC 1154 format.
Top	“Flatten” any message/rfc822 body part (forwarded message) into a header part and a message content part.
UUENCODE	Switch MIME encodings to X-UUENCODE.
Yes	Enable conversion.

13.6.1 Character Set Conversion

If the MTA probes and finds that the message is to be reformatted, it will proceed to check each part of the message. Any text parts are found and their character set parameters are used to

generate the second probe. Only when the MTA has checked and found that conversions may be needed does it ever perform the second probe. The input string in this second case looks like this:

`IN-CHAN=in-channel;OUT-CHAN=out-channel;IN-CHARSET=in-char-set`

The *in-channel* and *out-channel* are the same as before, and the *in-char-set* is the name of the character set associated with the particular part in question. If no match occurs for this second probe, no character set conversion is performed (although message reformatting, for example, changes to MIME structure, may be performed in accordance with the keyword matched on the first probe). If a match does occur it should produce a string of the form:

`OUT-CHARSET=out-char-set`

Here *out-char-set* specifies the name of the character set to which the *in-char-set* should be converted. Note that both of these character sets must be defined in the character set definition table, `charsets.txt`, located in the MTA table directory. No conversion will be done if the character sets are not properly defined in this file. This is not usually a problem since this file defines several hundred character sets; most of the character sets in use today are defined in this file. See the description of the `imsimta chbuild` (UNIX and NT) utility for further information on the `charsets.txt` file.

If all the conditions are met, the MTA will proceed to build the character set mapping and do the conversion. The converted message part will be relabelled with the name of the character set to which it was converted.

The charset-conversion mapping has been extended to provide several additional capabilities:

- A `IN-CHARSET` option can be specified in the output template of a mapping entry. If present this overrides the charset specified in the encoded-word.
- A `RELABEL-ONLY` option that accepts an integer 0 or 1 can be specified. If this option has a value of 1 the `OUT-CHARSET` simply replaces the `IN-CHARSET`; no relabelling is done.
- If the `IN-CHARSET` option is used to set the input charset to * the charset will be “sniffed” to determine an appropriate label.

EXAMPLE 13-2 Converting ISO-8859-1 to UTF-8 and back

Suppose that ISO-8859-1 is used locally, but this needs to be converted to UTF-8 for use on the Internet. In particular, suppose the connection to the Internet is via the `tcp_local` and `tcp_internal` and `ims-ms` are where internal messages originate and are delivered. The `CHARSET-CONVERSION` table shown below brings such conversions about. Note that each `IN-CHAN` entries must be on a single line. The backslash (\) is used to signify this.

CHARSET-CONVERSION

<code>IN-CHAN=tcp_internal;OUT-CHAN=tcp_local;CONVERT</code>	Yes
--	-----

EXAMPLE 13-2 Converting ISO-8859-1 to UTF-8 and back *(Continued)*

IN-CHAN=tcp_local;OUT-CHAN=tcp_internal;CONVERT	Yes
IN-CHAN=tcp_local;OUT-CHAN=ims-ms;CONVERT	Yes
IN-CHAN=*&;OUT-CHAN=*&;CONVERT	No
IN-CHAN=tcp_internal;OUT-CHAN=tcp_local;IN-CHARSET=ISO-8859-1 OUT-CHARSET=UTF-8	
IN-CHAN=tcp_local;OUT-CHAN=tcp_internal;IN-CHARSET=UTF-8 OUT-CHARSET=ISO-8859-1	
IN-CHAN=tcp_local;OUT-CHAN=ims-ms;IN-CHARSET=UTF-8	OUT-CHARSET=ISO-8859-1

EXAMPLE 13-3 Converting EUC-JP to ISO-2022-JP and Back

The CHARSET-CONVERSION table shown below specifies a conversion between local usage of EUC-JP and the ISO 2022 based JP code.

CHARSET-CONVERSION

IN-CHAN=ims-ms;OUT-CHAN=ims-ms;CONVERT	No
IN-CHAN=tcp_internal;OUT-CHAN=ims-ms;CONVERT	No
IN-CHAN=tcp_internal;OUT-CHAN=tcp_internal;CONVERT	No
IN-CHAN=tcp_internal;OUT-CHAN=*&;CONVERT	Yes
IN-CHAN=*&;OUT-CHAN=ims-ms;CONVERT	Yes
IN-CHAN=*&;OUT-CHAN=tcp_internal;CONVERT	Yes
IN-CHAN=tcp_internal;OUT-CHAN=*&;IN-CHARSET=EUC-JP	OUT-CHARSET=ISO-2022-JP
IN-CHAN=*&;OUT-CHAN=ims-ms;IN-CHARSET=ISO-2022-JP	OUT-CHARSET=EUC-JP
IN-CHAN=*&;OUT-CHAN=tcp_internal;IN-CHARSET=ISO-2022-JP	OUT-CHARSET=EUC-JP

13.6.2 Message Reformatting

As described above, the CHARSET-CONVERSION mapping table is also used to effect the conversion of attachments between MIME and several proprietary mail formats.

The following sections give examples of some of the other sorts of message reformatting which can be affected with the CHARSET-CONVERSION mapping table.

13.6.2.1

Non-MIME Binary Attachment Conversion

Mail in certain non-standard (non-MIME) formats; for example, mail in certain proprietary formats or mail from the Microsoft Mail (MSMAIL) SMTP gateway is automatically converted into MIME format if CHARSET-CONVERSION is enabled for any of the channels involved in handling the message. If you have a `tcp_local` channel then it is normally the incoming channel for messages from a Microsoft Mail SMTP gateway, and the following will enable the conversion of messages delivered to your local users:

CHARSET-CONVERSION

IN-CHAN=tcp_local;OUT-CHAN=ims-ms;CONVERT	Yes
---	-----

You may also wish to add entries for channels to other local mail systems. For instance, an entry for the `tcp_internal` channel:

CHARSET-CONVERSION

<code>IN-CHAN=tcp_local;OUT-CHAN=l;CONVERT</code>	Yes
<code>IN-CHAN=tcp_local;OUT-CHAN=tcp_internal;CONVERT</code>	Yes

Alternatively, to cover every channel you can simply specify `OUT-CHAN=*` instead of `OUT-CHAN=ims_ms`. However, this may bring about an increase in message processing overhead as all messages coming in the `tcp_local` channel will now be scrutinized instead of just those bound to specific channels.

More importantly, such indiscriminate conversions might place your system in the dubious and frowned upon position of converting messages—not necessarily your own site’s—which are merely passing through your system, a situation in which you should merely be acting as a transport and not necessarily altering anything beyond the message envelope and related transport information.

To convert MIME into the format Microsoft Mail SMTP gateway understands, use a separate channel in your MTA configuration for the Microsoft Mail SMTP gateway; for example, `tcp_msmail`, and put the following in the `mappings` file:

CHARSET-CONVERSION

<code>IN-CHAN=*,OUT-CHAN=tcp_msmail;CONVERT</code>	RFC1154
--	---------

13.6.2.2

Relabelling MIME Headers

Some user agents or gateways may emit messages with MIME headers that are less informative than they might be, but that nevertheless contain enough information to construct more precise MIME headers. Although the best solution is to properly configure such user agents or gateways, if they are not under your control, you can instead ask the MTA to try to reconstruct more useful MIME headers.

If the first probe of the `CHARSET-CONVERSION` mapping table yields a `Yes` or `Always` keyword, then the MTA will check for the presence of a `conversions` file. If a `conversions` file exists, then the MTA will look in it for an entry with `RELABEL=1` and if it finds such an entry, the MTA will then perform any MIME relabelling specified in the entry. See “[13.5.3 To Control Conversion Processing](#)” on page 435 for information on `conversions` file entries.

For example, the combination of a `CHARSET-CONVERSION` table such as:

CHARSET-CONVERSION

<code>IN-CHAN=tcp_local;OUT-CHAN=tcp_internal;CONVERT</code>	Yes
--	-----

and MTA `conversion` file entries of

```

out-chan=ims-ms; in-type=application; in-subtype=octet-stream;
in-parameter-name-0=name; in-parameter-value-0=*.ps;
out-type=application; out-subtype=postscript;
parameter-copy-0=*; relabel=1

out-chan=ims-ms; in-type=application; in-subtype=octet-stream;
in-parameter-name-0=name; in-parameter-value-0=*.msw;
out-type=application; out-subtype=msword;
parameter-copy-0=* relabel=1

```

will result in messages that arrive on the `tcp_local` channel and are routed to the `ims-ms` channel, and that arrive originally with MIME labelling of `application/octet-stream` but have a `filename` parameter with the extension `.ps` or `.msw`, being relabelled as `application/postscript` or `application/msword`, respectively. (Note that this more precise labelling is what the original user agent or gateway should have performed itself.) Such a relabelling can be particularly useful in conjunction with a `MIME-CONTENT-TYPES-TO-MR` mapping table, used to convert such resulting MIME types back into appropriate MRTYPE tags, which needs precise MIME labelling in order to function optimally; if all content types were left labelled only as `application/octet-stream`, the `MIME-CONTENT-TYPES-TO-MR` mapping table could only, at best, unconditionally convert all such to one sort of MRTYPE.

With the above example and `MIME-CONTENT-TYPES-TO-MR` mapping table entries including

APPLICATION/POSTSCRIPT	PS
APPLICATION/MSWORD	MW

a labelling coming in as, for example,

`Content-type: application/octet-stream; name=stuff.ps`

would be relabelled as

`Content-type: application/postscript`

and then converted into an MRTYPE tag `PS` to let Message Router know to expect PostScript.

Sometimes it is useful to do relabelling in the opposite sort of direction, “downgrading” specific MIME attachment labelling to `application/octet-stream`, the label for generic binary data. In particular, “downgrading” specific MIME labelling is often used in conjunction with the `convert_octet_stream` channel keyword on the `mime_to_x400` channel (PMDF-X400) or `xapi_local` channel (PMDF-MB400) to force all binary MIME attachments to be converted to X.400 bodypart 14 format.

For instance, the combination of a `CHARSET-CONVERSION` mapping table such as

`CHARSET-CONVERSION`

`IN-CHAN=*;OUT-CHAN=mime_to_x400*;CONVERT Yes`

and PMDF conversions file entries of

```
out-chan=mime_to_x400*; in-type=application; in-subtype=*
    out-type=application; out-subtype=octet-stream; relabel=1

out-chan=mime_to_x400*; in-type=audio; in-subtype=*
    out-type=application; out-subtype=octet-stream; relabel=1

out-chan=mime_to_x400*; in-type=image; in-subtype=*
    out-type=application; out-subtype=octet-stream; relabel=1

out-chan=mime_to_x400*; in-type=video; in-subtype=*
    out-type=application; out-subtype=octet-stream; relabel=1
```

will result in downgrading various specific MIME attachment labelling to the generic application/octet-stream labelling (so that `convert_octet_stream` will apply) for all messages going to `mime_to_x400*` channels.

13.6.2.3

MacMIME Format Conversions

Macintosh files have two parts, a resource fork that contains Macintosh specific information, and a data fork that contains data usable on other platforms. This introduces an additional complexity when transporting Macintosh files, as there are four different formats in common use for transporting the Macintosh file parts. Three of the formats, Applesingle, Binhex, and Macbinary, consist of the Macintosh resource fork and Macintosh data fork encoded together in one piece. The fourth format, Appledouble, is a multipart format with the resource fork and data fork in separate parts. Appledouble is hence the format most likely to be useful on non-Macintosh platforms, as in this case the resource fork part may be ignored and the data fork part is available for use by non-Macintosh applications. But the other formats may be useful when sending specifically to Macintoshes.

The MTA can convert between these various Macintosh formats. The `CHARSET-CONVERSION` keywords `Appledouble`, `Applesingle`, `Binhex`, or `Macbinary` tell the MTA to convert other MacMIME structured parts to a MIME structure of multipart/appledouble, application/applefile, application/mac-binhex40, or application/macbinary, respectively. Further, the `Binhex` or `Macbinary` keywords also request conversion to the specified format of non-MacMIME format parts that do nevertheless contain `X-MAC-TYPE` and `X-MAC-CREATOR` parameters on the MIME Content-type: header. The `CHARSET-CONVERSION` keyword `Block` tells the MTA to extract just the data fork from MacMIME format parts, discarding the resource fork; (since this loses information, use of `Appledouble` instead is generally preferable).

For example, the following `CHARSET-CONVERSION` table would tell the MTA to convert to Appledouble format when delivering to the VMS MAIL mailbox or a GroupWise postoffice, and to convert to Macbinary format when delivering to the Message Router channel:

<code>CHARSET-CONVERSION</code>	
<code>IN-CHAN=*</code> ; <code>OUT-CHAN=l</code> ; <code>CONVERT</code>	<code>Appledouble</code>

IN-CHAN=*;OUT-CHAN=wpo_local;CONVERT	Appledouble
IN-CHAN=*;OUT-CHAN=tcp_internal;CONVERT	Macbinary

The conversion to Appledouble format would only be applied to parts already in one of the MacMIME formats. The conversion to Macbinary format would only be applied to parts already in one of the MacMIME formats, or non-MacMIME parts which included X-MAC-TYPE and X-MAC-CREATOR parameters on the MIME Content-type: header.

When doing conversion to Appledouble or Block format, the MAC - TO - MIME - CONTENT - TYPES mapping table may be used to indicate what specific MIME label to put on the data fork of the Appledouble part, or the Block part, depending on what the Macintosh creator and Macintosh type information in the original Macintosh file were. Probes for this table have the form *format|type|creator|filename* where format is one of SINGLE, BINHEX or MACBINARY, where type and creator are the Macintosh type and Macintosh creator information in hex, respectively, and where filename is the filename.

For example, to convert to Appledouble when sending to the ims-ms channel and when doing so to use specific MIME labels for any MS Word or PostScript documents converted from MACBINARY or BINHEX parts, appropriate tables might be:

CHARSET - CONVERSION

IN-CHAN=*;OUT-CHAN=ims-ms;CONVERT	Appledouble
-----------------------------------	-------------

MAC - TO - MIME - CONTENT - TYPES

! PostScript	
MACBINARY 45505346 76677264 *	APPLICATION/POSTSCRIPT\$Y
BINHEX 45505346 76677264 *	APPLICATION/POSTSCRIPT\$Y
! Microsoft Word	
MACBINARY 5744424E 4D535744 *	APPLICATION/MSWORD\$Y
BINHEX 5744424E 4D535744 *	APPLICATION/MSWORD\$Y

Note that the template (right hand side) of the mapping entry must have the \$Y flag set in order for the specified labelling to be performed. Sample entries for additional types of attachments may be found in the file `mac_mappings.sample` in the MTA table directory.

If you wish to convert non-MacMIME format parts to Binhex or Macbinary format, such parts need to have X-MAC-TYPE and X-MAC-CREATOR MIME Content-type: parameter values provided. Note that MIME relabelling can be used to force such parameters onto parts that would not otherwise have them.

13.6.3 Service Conversions

The MTA's conversion service facility may be used to process with site-supplied procedures a message so as to produce a new form of the message. Unlike either the sorts of CHARSET-CONVERSION operations discussed above or the conversion channel, which operate on the content of individual MIME message parts, conversion services operate on entire MIME message parts (MIME headers and content) as well as entire MIME messages. Also, unlike other CHARSET-CONVERSION operations or conversion channel operations, conversion services are expected to do their own MIME disassembly, decoding, re-encoding, and reassembly.

Like other CHARSET-CONVERSION operations, conversion services are enabled through the CHARSET-CONVERSION mapping table. If the first probe of the CHARSET-CONVESION mapping table yields a Yes or Always keyword, then the MTA will check for the presence of an MTA conversions file. If a conversions file exists, then the MTA will look in it for an entry specifying a SERVICE-COMMAND, and if it finds such an entry, execute it. The conversions file entries should have the form:

```
in-chan=channel-pattern;
  in-type=type-pattern; in-subtype=subtype-pattern;
  service-command=command
```

Of key interest is the command string. This is the command that should be executed to perform a service conversion (for example, invoke a document converter). The command must process an input file containing the message text to be serviced and produce as output a file containing the new message text. On UNIX, the command must exit with a 0 if successful and a non-zero value otherwise.

For instance, the combination of a CHARSET-CONVERSION table such as

CHARSET-CONVERSION

IN-CHAN=bsout_* ; OUT-CHAN=* ; CONVERT Yes

and an MTA conversions file entry on UNIX of

```
in-chan=bsout_*; in-type=*; in-subtype=.*;
service-command="/pmdf/bin/compress.sh compress $INPUT_FILE $OUTPUT_FILE"
```

will result in all messages coming from a BSOUT channel being compressed.

Environment variables are used to pass the names of the input and output files as well as the name of a file containing the list of the message's envelope recipient addresses. The names of these environment variables are:

- INPUT_FILE - Name of the input file to process
- OUTPUT_FILE - Name of the output file to produce
- INFO_FILE - Name of the file containing envelope recipient addresses

The values of these three environment variables may be substituted into the command line by using standard command line substitution: that is, preceding the variable's name with a dollar character on UNIX. For example, when INPUT_FILE and OUTPUT_FILE have the values a.in and a.out, then the following declaration on UNIX:

```
in-chan=bsout_*; in-type=*; in-subtype=*
service-command="/pmdf/bin/convert.sh $INPUT_FILE $OUTPUT_FILE"
```

executes the command

```
/pmdf/bin/convert.sh a.in a.out
```


Integrating Spam and Virus Filtering Programs Into Messaging Server

This chapter describes how to integrate and configure spam and virus filtering software with Messaging Server. The spam/virus filtering technology described in this chapter is more powerful than the technology provided by the conversion channel (see “[13.5 The Conversion Channel](#)” on page 431). Messaging Server supports Symantec Brightmail AntiSpam, SpamAssassin, Milter, and anti-spam/anti-virus programs which support Internet Content Adaptation Protocol (ICAP, RFC 3507), specifically Symantec AntiVirus Scan Engine.

Note – In this chapter, references to *anti-spam* or *spam filtering* features also mean, when applicable, *anti-virus* or *virus filtering* features. Some products offer both (Brightmail), while others may offer only spam filtering (SpamAssassin) or only virus filtering (Symantec AntiVirus Scan Engine). Note also that *spam* is used generically in configuration parameters.

This chapter is divided into the following sections:

- “[14.1 Integrating Spam Filtering Programs Into Messaging Server—Theory of Operations](#)” on page 464
- “[14.2 Deploying and Configuring Third Party Spam Filtering Programs](#)” on page 464
- “[14.3 Using Symantec Brightmail Anti-Spam](#)” on page 476
- “[14.4 Using SpamAssassin](#)” on page 481
- “[14.5 Using Symantec Anti-Virus Scanning Engine \(SAVSE\)](#)” on page 493
- “[14.6 Using ClamAV](#)” on page 499
- “[14.7 Support for Sieve Extensions](#)” on page 504
- “[14.8 Using Milter](#)” on page 506
- “[14.9 Cloudmark Anti-Abuse Client](#)” on page 509
- “[14.10 Other Anti-Spam and Denial-of-Service Technologies](#)” on page 509

14.1 Integrating Spam Filtering Programs Into Messaging Server—Theory of Operations

From the perspective of Messaging Server, anti-spam solutions operate in much the same way:

1. Messaging Server sends a copy of a message to the spam filtering software.
2. The spam filtering software analyzes the message and returns a verdict of spam or not spam. Some programs, like SpamAssassin may also return a *spam score*, which is a numerical rating of the probability of the message being spam.
3. Messaging Server reads the verdict and takes a Sieve action on the message (see “[14.2.3 Specifying Actions to Perform on Spam Messages](#)” on page 471).

Spam filtering programs interact with the MTA through a protocol. The protocol may be a standard as in ICAP-based programs such as Symantec AntiVirus Scan Engine, proprietary as in Brightmail, or simply non-standard as in SpamAssassin. Each protocol requires software hooks to interface with the MTA. Brightmail and SpamAssassin were the first two spam filtering programs that could be integrated with messaging server. The MTA now supports the programs that use ICAP.

14.2 Deploying and Configuring Third Party Spam Filtering Programs

There are five actions required to deploy third-party filtering software on Messaging Server:

1. **Determine which spam filtering programs you wish to deploy, and how many servers on which to deploy them.** Messaging Server allows you to filter incoming messages with up to eight different spam/virus programs. These programs can be run on separate systems, on the same system as Messaging Server in a single system deployment, or on the same system as the MTA in a two-tier deployment. The number of servers required depends on the message load, the hardware performance, and other factors. Refer to your spam filtering software documentation or representative for guidelines on determining the hardware requirements at your site.
2. **Install and configure the spam filtering software.** Refer to your spam filtering software documentation or representative for this information.
3. **Load and configure the filtering client library.** This involves specifying the client libraries and configuration files in the MTA option.dat file, and also setting the desired options in the filtering software’s configuration files. See “[14.2.1 Loading and Configuring the Spam Filtering Software Client Library](#)” on page 465
4. **Specify what messages get filtered.** Messages can be filtered by user, domain, or channel. See “[14.2.2 Specifying the Messages to Be Filtered](#)” on page 466.

5. **Specify what happens to Spam.** Spam can be discarded, filed into a folder, tagged on the subject line, and so on. See “[14.2.3 Specifying Actions to Perform on Spam Messages](#)” on page 471

Note – Previous versions of Messaging Server only supported the Brightmail filtering technology and so keywords and options had names, such as `sourcebrightmail` or `Brightmail_config_file`. These keywords and options have been changed to more generic names such as `sourcespamfilter` or `spamfilter_config_file`. The previous Brightmail names are retained for compatibility.

14.2.1 Loading and Configuring the Spam Filtering Software Client Library

Each spam filtering program is expected to provide a client library file and configuration file for Messaging Server. Loading and configuring the client library involves two things:

- Specifying the spam filtering software library path (`spamfilterX_library`) and configuration file (`spamfilterX_config_file`) in the `option.dat` file. In addition to these options, there are a number of others used to specify spam filtering LDAP attributes, as well as the Sieve actions to use on spam messages.
- Specifying the desired options in the spam filtering software configuration files. Each spam filtering program has a different configuration file and configuration options. These are described in the spam filtering software sections, as well as in the filtering software documentation. See “[14.3 Using Symantec Brightmail Anti-Spam](#)” on page 476 and “[14.5 Using Symantec Anti-Virus Scanning Engine \(SAVSE\)](#)” on page 493

14.2.1.1 Specifying the Spam Filtering Software Library Paths

Messaging Server can call up to eight different filtering systems for your messages. For example, you can run your messages through both the Symantec AntiVirus Scan Engine and SpamAssassin. Each filtering software is identified by a number from 1 to 8. These numbers appear as part of the various spam filter options, LDAP attributes, and channel keywords; an X is used as a filter identification number. For example, `sourcespamfilterXoptin` or `spamfilterX_config_file`. If the identifying number is omitted from the keyword or option name it defaults to 1.

The following `option.dat` settings specify Messaging Server to filter messages through both Symantec AntiVirus Scan Engine and SpamAssassin:

```
spamfilter1_library=Symantec_Library_File  
spamfilter1_config_file=Symantec_Config_File  
spamfilter2_library=SpamAssassin_Library_File  
spamfilter2_config_file=SpamAssassin_Config_File
```

When using other options or keywords to configure the system, use the corresponding number at the end of the option or keyword. For example, `sourcespamfilter2optin` would refer to SpamAssassin. `sourcespamfilter1optin` would refer to Symantec AntiVirus Scan Engine. It is not necessary to use numbers sequentially. For example, if you want to temporarily disable the Symantec AntiVirus Scan Engine, you can just comment out the `spamfilter1_library` configuration file.

14.2.2

Specifying the Messages to Be Filtered

Once the spam filtering software is installed and ready to run with Messaging Server, you need to specify what messages to filter. Messaging Server can be configured to filter messages by user, domain, or channel. Each of these scenarios is described in the following sections:

- “[To Specify User-level Filtering](#)” on page 466
- “[14.2.2.1 User-level Filtering Example](#)” on page 467

Note – The expression *optin* means that a user, domain or channel is selected to receive mail filtering.

▼ To Specify User-level Filtering

It may be desirable to specify filtering on a per-user basis. For example, if spam or virus filtering is offered as a premium service to ISP customers, you can specify which users receive this and which don’t. The general steps for user filtering are as follows:

1 Specify the user LDAP attributes that activate the spam filtering software.

Set the `LDAP_OPTINX` options in `option.dat`. Example:

```
LDAP_OPTIN1=SymantecAV  
LDAP_OPTIN2=SpamAssassin
```

Note – By default, the attributes like `SymantecAV` or `SpamAssassin` do not exist in the schema. Whatever new attributes you use, you will need to add them to your directory schema. See the appropriate Directory Server documentation for instructions.

2 Set filter attributes in the user entries that receive spam filtering.

The values for the filter attributes are multi-valued and depend on the server. Using the example shown in Step 1, the entries are:

```
SymantecAV: virus  
SpamAssassin: spam
```

For a program like Brightmail, which can filter both viruses and spam, the valid values are `spam` and `virus`. When used as a multi-valued attribute, each value requires a separate attribute entry. For example, if the filter attribute for Brightmail was set to Brightmail, the entries are:

```
Brightmail: spam  
Brightmail: virus
```

14.2.2.1 User-level Filtering Example

This example assumes that Brightmail is used. It also assumes that `LDAP_OPTIN1` was set to Brightmail in the `option.dat` file. The user, Otis Fanning, has the Brightmail attribute set to `spam` and `virus` in his user entry. His mail is filtered by Brightmail for spam and viruses. “[14.2.2.1 User-level Filtering Example](#)” on page 467 shows the Brightmail user entry for Otis Fanning.

EXAMPLE 14-1 Example LDAP User Entry for Brightmail

```
dn: uid=fanning,ou=people,o=sesta.com,o=ISP  
objectClass: person  
objectClass: organizationalPerson  
objectClass: inetOrgPerson  
objectClass: inetUser  
objectClass: ipUser  
objectClass: inetMailUser  
objectClass: inetLocalMailRecipient  
objectClass: nsManagedPerson  
objectClass: userPresenceProfile  
cn: Otis Fanning  
sn: fanning  
initials: OTF  
givenName: Otis  
pabURI: ldap://ldap.siroe.com:389/ou=fanning,ou=people,o=sesta.com,o=isp,o=pab  
mail: Otis.Fanning@sesta.com  
mailAlternateAddress: ofanning@sesta.com  
mailDeliveryOption: mailbox  
mailHost: manatee.siroe.com  
uid: fanning  
dataSource: iMS 5.0 @(#)ims50users.sh 1.5a 02/3/00  
userPassword: password  
inetUserStatus: active  
mailUserStatus: active  
mailQuota: -1  
mailMsgQuota: 100  
Brightmail: virus  
Brightmail: spam
```

If Symantec AntiVirus Scan Engine and SpamAssassin were used, the entry would look like this:

SymantecAV: virus
SpamAssassin: spam

See “[14.3 Using Symantec Brightmail Anti-Spam](#)” on page 476, “[14.4 Using SpamAssassin](#)” on page 481 or “[14.5 Using Symantec Anti-Virus Scanning Engine \(SAVSE\)](#)” on page 493

▼ To Specify Domain-level Filtering

You can specify which domains receive filtering. An example of this feature would be if anti-spam or anti-virus filtering were offered as a premium service to ISP domain customers. The general steps for specifying domain filtering is as follows:

1 Specify the domain LDAP attributes that activates the filtering software.

Set the LDAP_DOMAIN_ATTR_OPTINX options in option.dat. Example:

LDAP_DOMAIN_ATTR_OPTIN1=SymantecAV
LDAP_DOMAIN_ATTR_OPTIN2=SpamAssassin

Note – By default, the attributes like SymantecAV or SpamAssassin do not exist in the schema. Whatever new attributes you use, you will need to add them to your directory schema. See the appropriate Directory Server documentation for instructions.

2 Set filter attributes in the domain entries that receive spam filtering.

The values for the filter attributes are multi-valued and depend on the server. Using the example shown in Step 1, the entries would be as follows:

SymantecAV: virus
SpamAssassin: spam

For a program like Brightmail which can filter both viruses and spam, the valid values are spam and virus. When used as a multi-valued attribute, each value requires a separate attribute value entry. For example, if LDAP_DOMAIN_ATTR_OPTIN1 was set to Brightmail, the entries would be:

Brightmail: spam
Brightmail: virus

Domain-level Filtering Example

This example assumes that Brightmail is used. It also assumes that LDAP_DOMAIN_ATTR_OPTIN1 was set to Brightmail in the option.dat file. The Brightmail attribute is set to spam and virus in the sesta.com domain entry in the DC tree for Sun LDAP Schema 1. For Sun LDAP Schema 2 you also set Brightmail in the domain entries that receive spam filtering.

All mail sent to sesta.com is filtered for spam and viruses by Brightmail. A “[Domain-level Filtering Example](#)” on page 468 is shown below.

EXAMPLE 14-2 Example LDAP Domain Entry for Brightmail

```
dn: dc=sesta,dc=com,o=internet
objectClass: domain
objectClass: inetDomain
objectClass: mailDomain
objectClass: nsManagedDomain
objectClass: icsCalendarDomain
description: DC node for sesta.com hosted domain
dc: sesta
inetDomainBaseDN: o=sesta.com,o=isp
inetDomainStatus: active
mailDomainStatus: active
mailDomainAllowedServiceAccess: +imap, pop3, http:*
mailRoutingHosts: manatee.siroe.com
preferredMailHost: manatee.siroe.com
mailDomainDiskQuota: 100000000
mailDomainMsgQuota: -1
mailClientAttachmentQuota: 5
Brightmail: spam
Brightmail: virus
```

If Symantec AntiVirus Scan Engine and SpamAssassin were used, the entry would look similar to like this:

```
SymantecAV: virus
SpamAssassin: spam
```

See “[14.3 Using Symantec Brightmail Anti-Spam](#)” on page 476, “[14.4 Using SpamAssassin](#)” on page 481 or “[14.5 Using Symantec Anti-Virus Scanning Engine \(SAVSE\)](#)” on page 493 for more examples and details.

▼ To Specify Channel-level Filtering

Filtering by source or destination channel provides greater flexibility and granularity for spam filtering. For example, you may wish to filter in these ways:

- Only messages from a specific MTA relay to a backend message store
- All incoming mail from a specific MTA.
- All outgoing mail from a specific MTA.
- Incoming and outgoing mail from a specific MTA.

Messaging Server allows you to specify filtering by source or destination channel. The mechanism for doing this are the channel keywords described in “[12.12.5 Spam Filter Keywords](#)” on page 420. The following example demonstrates how to set up channel-level filtering.

- 1 Add a rewrite rule in the `imta.cnf` file for all inbound SMTP servers that send messages to a backend message store host. Example:

```
msg_store1.siroe.com $U@msg_store1.siroe.com
```

- 2 Add a channel corresponding to the rewrite rule with the `destinationspamfilterXoptin` keyword. Example:

```
tcp_msg_store1 smtp subdirs 20 backoff "pt5m" "pt10" "pt30" \
"pt1h" "pt2h" "pt4h" maxjobs 1 pool IMS_POOL \
fileinto $U+$S@$D destinationspamfilterloptin spam
msg_store1.siroe.com
```

Channel-level Filtering Examples

These examples assume a filtering program specified by the number 1. See “[12.12.5 Spam Filter Keywords](#)” on page [420](#) for the keywords available for spam filtering.

▼ To Filter from an MTA Relay to a Backend Message Store

This example filters all mail for spam and viruses from an MTA relay to a backend message store called `msg_store1.siroe.com`

- 1 Add a rewrite rule in the `imta.cnf` file that sends messages to a backend message store host. Example:

```
msg_store1.siroe.com $U@msg_store1.siroe.com
```

- 2 Add a channel corresponding to that rewrite rule with the `destinationspamfilterXoptin` keyword. Example:

```
tcp_msg_store1 smtp subdirs 20 backoff "pt5m" "pt10" "pt30" "pt1h" \
"pt2h" "pt4h" maxjobs 1 pool IMS_POOL fileinto $U+$S@$D \
destinationspamfilter loptin spam,virus
msg_store1.siroe.com
```

Example 2. Filter for spam all incoming mail passing through your MTA (Typically, all incoming messages pass through the `tcp_local` channel):

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonlimited subdirs 20 maxjobs 7 pool SMTP_POOL \
maytlsserver mayaslserver saslswitchchannel tcp_auth \
sourcesspamfilterloptin spam
tcp-daemon
```

Example 3. Filter all outgoing mail to the Internet passing through your MTA. (Typically, all messages going out to the Internet pass through the `tcp_local` channel.)

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonlimited subdirs 20 maxjobs 7 pool SMTP_POOL \
```

```
maytlsserver maysaslserver saslswitchchannel tcp_auth \
destinationspamfilterloptin spam tcp-daemon
```

Example 4. Filter all incoming and outgoing mail passing through your MTA:

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL \
maytlsserver maysaslserver saslswitchchannel tcp_auth \
sourcespamfilterloptin spam destinationspamfilterloptin spam
tcp-daemon
```

Example 5. Filter all mail destined to the local message store in a two-tiered system without using user optin:

```
ims-ms smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL \
maytlsserver maysaslserver saslswitchchannel tcp_auth \
destinationspamfilterloptin spam
tcp-daemon
```

Example 6. Filter all incoming and outgoing mail for spam and viruses (this presumes that your software filters both spam and viruses):

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL \
maytlsserver maysaslserver saslswitchchannel tcp_auth \
destinationspamfilterloptin spam,virus sourcespamfilterloptin \spam,virus
tcp-daemon
```

14.2.3

Specifying Actions to Perform on Spam Messages

Spam filtering programs analyze messages and return a verdict of spam or not spam to current version of Messaging Server. Messaging Server then takes action on the message. Actions are specified using the Sieve mail filtering language. Possible actions are to discard the message, file it into a folder, add a header, add a tag to the subject line, and so on. Complex Sieve scripts with if-then-else statements are also possible.

Note – Refer to the Sieve specification 3028 for the complete Sieve syntax. Also see
<http://www.cyrussoft.com/sievettpp://www.cyrussoft.com/sieve/>
[\(http://www.cyrussoft.com/sieve/\)](http://www.cyrussoft.com/sieve/)

Sieve scripts are specified with the MTA spam filter options (`option.dat`) described in [Table 14-1](#). The primary spam filter action options are `SpamfilterX_null_action`, which specifies the Sieve rule to execute when a null value is returned as the spam verdict value, and `SpamfilterX_string_action`, which specifies the Sieve rule to execute when a string is returned as the spam verdict.

Spam filtering programs typically return a string or a null value to the MTA to indicate that message is spam. Some programs also return a spam score—a number rating the probability of the message being spam or not. This score can be used as part of the action sequence. The following examples show how to specify actions on filtered messages. Each example assumes a filtering program specified by the number 1.

Example 1: File spam messages with a null verdict value to the file SPAM_CAN.

```
spamfilter1_null_action=data:,require "fileinto"; fileinto "SPAM_CAN";
```

The same action can be performed on a spam message that returns a string:

```
spamfilter1_string_action=data:,require "fileinto"; fileinto "SPAM_CAN";
```

Example 2: File spam messages with a returned verdict string into a file named after the returned verdict string (this is what \$U does). That is, if the verdict string returned is spam, the message is stored in a file called spam.

```
spamfilter1_null_action=data:,require "fileinto"; fileinto "$U";
```

Example 3: Discard spam messages with a string verdict value.

```
spamfilter1_string_action=data:,discard
```

The same action can be performed on a spam message that returns a null value:

```
spamfilter1_null_action=data:,require "fileinto"; fileinto "SPAM_CAN";
```

Example 4. This line adds the header Spam-test: FAIL to each message determined to be spam by a string verdict value:

```
spamfilter1_string_action=data:,require ["addheader"]; addheader "Spam-test: FAIL";
```

Example 5. This line adds the string [PROBABLE SPAM] to the subject line of the spam messages returning a string:

```
spamfilter1_string_action=data:,addtag "[PROBABLE SPAM]";
```

Example 6. This line assumes a string verdict value and files a spam message in the mailbox testspam if the header contains resent-from and User-1. If the message does not have that header, it files the message into spam.

```
spamfilter1_string_action=data:,require "fileinto"; \
  if header :contains ["resent-from"] ["User-1"] { \
    fileinto "testspam"; \
  } else { \
    fileinto "spam";};
```

Because verdict strings are configurable with most spam filter software, you can specify different actions depending on the returned string. This can be done with the matched pairs `spamfilterX_verdict_n` and `spamfilterX_action_n` options.

Example 7. These matched pair options discard spam messages with the returned verdict string of remove.

```
spamfilter1_verdict_0=remove
spamfilter1_action_0=data:,discard
```

Refer to the specific spam filtering software sections for instructions on how to specify the spam verdict string.

TABLE 14-1 MTA Spam Filter Options (option.dat)

MTA Options	Description
<code>SpamfilterX_config_file</code>	Specifies the full file path and name of the filtering software X configuration file. Default: none
<code>SpamfilterX_library</code>	Specifies the full file path and name of the filtering software X shared library. Default: none
<code>SpamfilterX_optional</code>	Controls whether certain failures reported by the filtering library X are treated as a temporary processing failure or ignored. 0 specifies that spam filtering problems cause a temporary processing failure. 1 causes spam filter processing to be skipped in the event of some, but possibly not all, filtering library failures. In particular, if the system gets stuck without a return in the library code, some portion of the MTA may also get stuck. -2 and 2 are the same as 0 and 1 respectively except that they also cause a syslog message to be sent in the event of a problem reported by the spam filter plugin. 3 causes spam filter failures to accept the message, but queue it to the reprocess channel for later processing. 4 does the same as 3 but also logs the spam filter temporary failure to syslog. Default: 0
<code>LDAP_optinX</code>	Specifies the name of the LDAP attribute used to activate filtering software X on a per-user basis. Filtering is based on destination addresses. That is, messages directed to users with this attribute will be filtered for spam. This should be an attribute in the <code>inetMailUser</code> objectclass. The attribute itself can take multiple values and is case-sensitive. For SpamAssassin, its value should be <code>spam</code> in lowercase. Default: none
<code>LDAP_SOURCE_OPTINX</code>	LDAP_SOURCE_OPTIN1 through LDAP_SOURCE_OPTIN8 provide originator-address-based per-user spam filter optins values comparable to <code>LDAP_optinX</code> . This is, mail originating from this user will be filtered for spam.

TABLE 14–1 MTA Spam Filter Options (option.dat) *(Continued)*

MTA Options	Description
LDAP_domain_attr_optinX	Specifies the name of the LDAP attribute used to activate filtering software X on a domain basis. It applies to the destination domain. It is just like LDAP_optin, except it should be in the objectclass mailDomain. Default: none
SpamfilterX_null_optin	Specifies a string which, if found, as a value of the attribute defined by LDAP_optinX or LDAP_domain_attr_optinX, causes the MTA to act as if the attribute wasn't there. That is, it disables filtering for that entry. See “ 14.2.2 Specifying the Messages to Be Filtered ” on page 466 Default: The empty string. Empty optin attributes are ignored by default. (This is a change from iPlanet Messaging Server 5.2, where empty optin attributes triggered filtering with an empty optin list. The 5.2 behavior can be restored by setting spamfilterX_null_optin to a string that never occurs in practice.)
SpamfilterX_null_action	Defines a Sieve rule specifying what to do with the message when the filtering software X verdict returns as null. Sieve expressions can be stored externally using a file URL. For example: file:///var/opt/SUNWmsgsr/config/null_action.sieve. Also, do not reject spam using the Sieve reject action, as it tends to deliver a nondelivery notification to the innocent party whose address was used to send the spam. Default: data:,discard;
SpamfilterX_string_action	Defines Sieve rule specifying what to do with the message if the verdict is a string. Sieve expressions can be stored externally using a file URL. For example: file:///var/opt/SUNWmsgsr/config/null_action.sieve. Also, do not reject spam using the Sieve reject action, as it tends to deliver a nondelivery notification to the innocent party whose server was used to send the spam. Default: data:,require "fileinto"; fileinto "\$U; where \$U is the string that verdict returned.
spamfilterX_verdict_n	The options spamfilterX_verdict_n and spamfilterX_action_n are matched pairs, where n is a number from 0 to 9. These options allow you to specify Sieve filters for arbitrary verdict strings. This is done by setting spamfilterX_verdict_n and spamfilterX_action_n to the verdict string and sieve filter, respectively, where n is an integer from 0 to 9. For example, a site could have the “reject” verdict cause a sieve reject action by specifying: <code>spamfilter1_verdict_0=reject spamfilter1_action_0=data:,require "reject"; reject "Rejected by spam filter";</code> The default values for all the spamfilterX_verdict_n options and the corresponding action options are empty strings. Default: none
spamfilterX_action_n	See spamfilterX_verdict_n. Default: none

TABLE 14-1 MTA Spam Filter Options (option.dat) *(Continued)*

MTA Options	Description
<code>spamfilterX_final</code>	<p>Some filtering libraries have the ability to perform a set of actions based on recipient addresses. What sort of recipient address is passed to the filtering library depends on the setting of the <code>spamfilterX_final</code>. The default value of 0 results in a so-called intermediate address being passed to the filtering library. This address is suitable for use in delivery status notifications and for directory lookups. If bit 0 (value 1) of <code>spamfilterX_final</code> is set, however, the final form of the recipient address is passed. This form may not be suitable for presentation but is more appropriate for subsequent forwarding operations.</p> <p>The <code>spamfilterX_final</code> option is only available in Messaging Server 6.0; Messaging Server 5.2 behaves as if the option had the default value of 0.</p> <p>In 6.2 and later bit 1 (value 2) of <code>spamfilterX_final</code> controls whether or not source routes are stripped from the address that's passed to the filtering interface. Setting the bit enables source route stripping.</p> <p>Default: 0</p>
<code>optin_user_carryover</code>	<p>Forwarding is a challenge for spam filter processing. Consider a user entry that specifies the <code>forward</code> delivery option and specifies the forwarding address of another user. Additionally, the user entry is set to opt in to some specific sort of filtering. Should the filtering be applied to the forwarded message or not? On the one hand, the correct filtering choice for one particular user may not be the correct choice for another. On the other hand, eliminating a filtering operation might be used as means of violating a site's security policy.</p> <p>No single answer is correct in all cases so <code>OPTIN_USER_CARRYOVER</code> controls how the spam filtering optin list is carried from one user or alias entry to another when forwarding occurs. This is a bit-encoded value. The various bit values have the following meanings:</p> <ul style="list-style-type: none"> bit 0 (value 1). Each LDAP user entry overrides any previously active user/domain optins unconditionally. bit 1 (value 2). If a user's domain has an optin attribute, it overrides any previous user/domain/alias optins that were active. bit 2 (value 4). If a user has an optin attribute, it overrides any previous user/domain/alias optins that were active. bit 3 (value 8). An optin specified by an [optin] non-positional parameter overrides any previous user/domain/alias optins that were active. <p>Default: 0 (optins accumulate if one user has a delivery option that forwards to another. The default insures that site security policies are effective when forwarding; other settings may not.)</p>

14.3 Using Symantec Brightmail Anti-Spam

The Brightmail solution consists of the Brightmail server along with realtime anti-spam and anti-virus rule updates downloaded to email servers. In addition to the sections below, refer to *Configuring Brightmail with Sun Java System Messaging Server*.

- “14.3.1 How Brightmail Works” on page 476
- “14.3.2 Brightmail Requirements and Performance Considerations” on page 478
- “14.3.3 Deploying Brightmail” on page 479
- “14.3.4 Brightmail Configuration Options” on page 479

14.3.1 How Brightmail Works

The Brightmail server is deployed at a customer site. Brightmail has email probes set around the Internet for detection of new spam. Brightmail technicians create custom rules to block this spam in realtime. These rules are downloaded to Brightmail servers, also in realtime. The Brightmail database is updated and Brightmail server runs this database filter against the email for the specified users or domains.

14.3.1.1 Brightmail Architecture

Figure 14–1 depicts the Brightmail architecture.

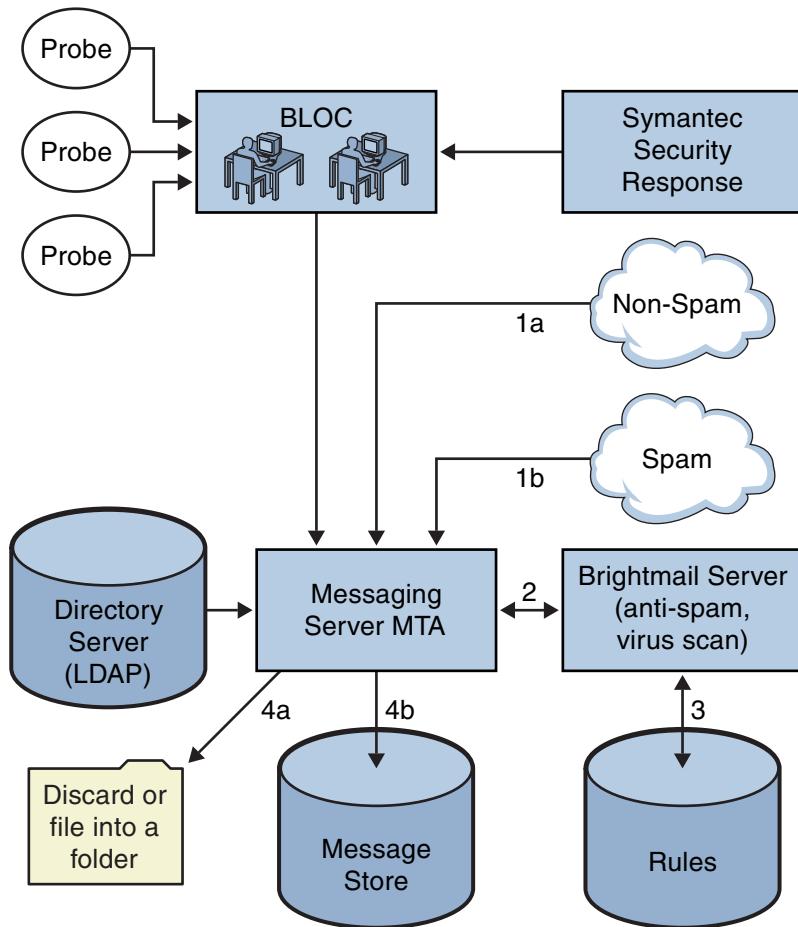


FIGURE 14-1 Brightmail and Messaging Server Architecture

When the Brightmail Logistics and Operations Center (BLOC) receives spam from email probes, operators immediately create appropriate spam filtering rules, which are downloaded to Brightmail customer machines. Similarly, the Symantec Security Response realtime virus rules are also sent from Brightmail. These rules are used by customer's Brightmail servers to catch spam and viruses.

The MTA uses the Brightmail SDK to communicate with the Brightmail Server. The MTA dispatches messages based on the response from Brightmail. After the mail (1a) or (1b) is received by the MTA, the MTA sends the message to the Brightmail server (2). The Brightmail server uses its rules and data to determine if the message is a spam or virus (3), and returns a verdict to the MTA. Based on the verdict, the MTA either (4a) discards the message or files the message into a folder, or (4b) delivers it normally to the destination.

Since the Brightmail SDK is third party software, we do not include it in our installation kit. The Brightmail SDK and server software must be obtained from Brightmail Inc. The MTA has configuration settings to tell it whether and where to load the Brightmail SDK to enable Brightmail integration.

Once the SDK is loaded, Brightmail message processing is determined by several factors and levels of granularity (the term used by Brightmail to specify active processing is *optin*). This is specified by the following criteria:

- Whether the source or destination channel is enabled for Brightmail (`imta.cnf`)
- Whether there is a channel default for the services opted in (`imta.cnf`)
- Whether there is a per domain optin (LDAP)
- Whether there is per-user optin (LDAP)

For any particular message recipient, the optins and defaults above are combined, which means, if the channel default is already specified for both spam and virus, then there is no reason to bother with per-user optin. That is, if the system administrator decides to do spam and virus filtering for everyone, then there is no reason to expose to the user the ability to optin for spam or virus. There is no way to opt out of processing, that is, you can not say you do not want the service if a user is already optin via a system or domain optin. This also means that if you are optin for a service, and you have forwarded your mail to another address, that address would get the mail after the filtering has been performed on your behalf.

There are only two services offered, virus or spam detection. Brightmail also provides “content-filtering” service, but this functionality is provided using Sieve, so there is no added value to have Brightmail do the Sieve filtering.

When a message is determined to contain a virus, the Brightmail server can be configured to clean the virus and resubmit the cleaned message back to the MTA. (Due to some undesirable side effects caused by loss of information about the original message in a resubmitted cleaned message, we recommend you do not configure Brightmail to resubmit the cleaned message back to the MTA.) When the message is spam, the verdict back from the Brightmail along with the configuration in Brightmail allows the MTA to determine what happens to the message. The message can be discarded, filed into a folder, tagged as spam or virus on the subject line, passed to a Sieve rule, delivered normally in the INBOX, and so on.

The Brightmail servers can be located on the same system as the MTA, or it can be on a separate system. In fact, you can have a farm of Brightmail servers serving one or more MTAs. The Brightmail SDK uses the Brightmail configuration file to determine which Brightmail servers to use.

14.3.2 Brightmail Requirements and Performance Considerations

- Brightmail servers must run on the Solaris Operating System.

- If Brightmail does spam and virus checking, MTA message throughput can be reduced by as much 50%. To keep up with MTA throughput, you may need two Brightmail servers for each MTA.
- While SpamAssassin has the ability to perform different sorts of filtering on a user basis, it is unable to apply two different sets of filtering criteria to the same message at the same time. Thus, SpamAssassin only allows system-wide filtering. Customized filtering for individual users is not possible.

14.3.3 Deploying Brightmail

Perform the following steps to deploy Brightmail.

- **Install and Configure Brightmail.** Refer to the Brightmail software documentation or representative for installation and configuration information. Selected Brightmail configuration options are shown in “[14.3.4 Brightmail Configuration Options](#)” on page 479, however the most complete and up-to-date information is in the Brightmail documentation.
- **Load and Configure the Brightmail Client Library.** This involves specifying the Brightmail client library, `libbmclient.so`, and configuration, `config`, file to the MTA. See “[14.2.1 Loading and Configuring the Spam Filtering Software Client Library](#)” on page 465
- **Specify what messages to filter for spam.** Messages can be filtered by user, domain, or channel. See “[14.2.2 Specifying the Messages to Be Filtered](#)” on page 466
- **Specify what actions to take on spam messages.** Spam can be discarded, filed into a folder, tagged on the subject line, and so on. See “[14.2.3 Specifying Actions to Perform on Spam Messages](#)” on page 471
- **Set miscellaneous MTA filter configuration parameters as desired.** See [Table 14–1](#).

14.3.4 Brightmail Configuration Options

Selected Brightmail configuration file options are shown in [Table 14–2](#). The most complete listing of Brightmail configuration file options can be obtained from Brightmail. Options and values are not case-sensitive.

TABLE 14–2 Selected Brightmail Configuration File Options

Brightmail Option	Description
<code>b1SWPrecedence</code>	A given message can have multiple verdicts. This specifies the precedence order. So if a message is processed for virus first, then for spam if you specified this option as <code>virus-spam</code> the verdicts are separated by hyphens (-). This is the recommended setting when using Brightmail with Sun Java System Messaging Server.

TABLE 14–2 Selected Brightmail Configuration File Options *(Continued)*

Brightmail Option	Description
<code>b1SWClientDestinationDefault</code>	Specifies how to deliver normal messages, that is, not a spam or virus, and thus have no verdict. Usually you want to deliver this message normally, so you would specify <code>inbox</code> as the value. There is no default.
<code>b1SWLocalDomain</code>	This attribute specifies what domain(s) are considered to be local. There can be multiple lines of this attribute specifying several domains which are all considered local. Local versus foreign domain is used to specify two different handling for a verdict. See below <code>b1SWClientDestinationLocal</code> and <code>b1SWClientDestinationForeign</code> . For example, you can specify <code>b1SWLocalDomain=siroe.com</code>
<code>b1SWClientDestinationLocal</code>	This specifies the verdict and action pair for the local domain. You would normally have two lines for this, one for spam and one for virus. The value is of the form <code>verdict action</code> . For example, <code>b1SWClientDestinationLocal=spam spambox</code> <code>b1SWClientDestinationLocal=virus </code> The default Brightmail interpretation for the “null” action, meaning nothing to the right of the <code> </code> , is to discard the message. So the example above discards the message if it has a verdict of <code>virus</code> . And if the verdict is <code>spam</code> , the above example files the message into the folder called <code>spambox</code> . If the message is not spam or virus, then the verdicts do not match, and the mail is delivered normally based on what’s set in the <code>b1SWClientDestinationDefault</code> setting above. When using a separate Brightmail server or servers from the MTA, you can customize the actions taken by each MTA by using the <code>Brightmail_verdict_n</code> , <code>Brightmail_action_n</code> , <code>Brightmail_null_action</code> , and <code>Brightmail_string_action</code> MTA options to override the actions and verdicts returned by the Brightmail server. In this example, you can use different <code>Brightmail_null_action</code> on the MTA to override the Virus action (which would be to discard it) or to use <code>Brightmail_verdict_0=spambox</code> , and <code>Brightmail_action_0=data:,require "fileinto";fileinto "Junk"</code> ; to file into a folder called <code>Junk</code> instead of <code>spambox</code> .
<code>b1SWClientDesintationForeign</code>	Same format and interpretation as <code>b1SWClientDestinationLocal</code> above, except this applies to users in the domain which are NOT local.
<code>b1SWUseClientOptin</code>	Always set this to TRUE when used with Sun Java System Messaging Server.
<code>b1swcServerAddress</code>	Is of the form <code>ip:port[,ip:port,...]</code> to specify one or more Brightmail server’s IP address and port numbers

14.4 Using SpamAssassin

This section consists of the following subsections:

- “14.4.1 SpamAssassin Overview” on page 481
- “14.4.2 SpamAssassin/Messaging Server Theory of Operations” on page 481
- “14.4.3 SpamAssassin Requirements and Usage Considerations” on page 482
- “14.4.4 Deploying SpamAssassin” on page 483
- “14.4.5 SpamAssassin Configuration Examples” on page 483
- “14.4.6 Testing SpamAssassin” on page 489
- “14.4.7 SpamAssassin Options” on page 491

14.4.1 SpamAssassin Overview

Messaging Server supports the use of SpamAssassin, a freeware mail filter used to identify spam. SpamAssassin consists of a library written in Perl and a set of applications and utilities that can be used to integrate SpamAssassin into messaging systems.

SpamAssassin calculates a score for every message by performing a series of tests on the message header and body information. Each test succeeds or fails, and a verdict of true (spam) or false (not spam) is rendered. Scores are real numbers that can be positive or negative. Scores that exceed a specified threshold, typically 5.0, are considered to be spam. An example of a SpamAssassin result string is:

True ; 18.3 / 5.0

True indicates the message is spam. 18.3 is the SpamAssassin score. 5.0 is the threshold.

SpamAssassin is highly configurable. Tests may be added or removed at any time, and the scores of existing tests can be adjusted. This is all done through various configuration files. Further information on SpamAssassin can be found on the SpamAssassin web site.

The same mechanism used for calling out to the Brightmail spam and virus scanning library can be used to connect to the SpamAssassin `spamd` server. The module provided in Messaging Server is called `libspamass.so`.

14.4.2 SpamAssassin/Messaging Server Theory of Operations

`spamd` is the daemon version of SpamAssassin and can be invoked from the MTA. `spamd` listens on a socket for requests and spawns a child process to test the message. The child process dies after processing the message and sending back a result. In theory, the forking should be an efficient process because the code itself is shared among the children processes.

The client portion, `spamc` from the SpamAssassin installation, is not used. Instead, its function is done by a shared library called `libspamass.so`, which is part of the Messaging Server. `libspamass.so` is loaded the same way that the Brightmail SDK is loaded.

From the MTA's point of view, you can almost transparently switch between SpamAssassin and Brightmail for spam filtering. It's not completely transparent, because they do not have the same functions. For example, Brightmail can also filter for viruses, but SpamAssassin is only used to filter for spam. The result, or *verdict*, returned by the two software packages is also different. SpamAssassin provides a score, while Brightmail provides just the verdict name, so the configuration would also have some differences.

When using SpamAssassin with the MTA, only a score and verdict is returned from SpamAssassin. The message itself is not modified. That is, options such as adding headers and modifying subject lines must be done by Sieve scripts. In addition, the `mode` option allows you to specify the string that is returned to indicate the verdict. The string choices are null, default, SpamAssassin result string, or a verdict string. See “[14.4.7 SpamAssassin Options](#)” on page 491 for details.

14.4.3 SpamAssassin Requirements and Usage Considerations

- SpamAssassin is free. Go to <http://www.spamassassin.org> for software and documentation.
- SpamAssassin can be tuned and configured to provide very accurate detection of spam. The tuning is up to you and the SpamAssassin community. Messaging Server does not provide or enhance what SpamAssassin can do.
- While no specific numbers are available, SpamAssassin seems to reduce throughput more than Brightmail.
- SpamAssassin integrated with the MTA can be enabled for a user, a domain, or a channel.
- SpamAssassin can be configured to use other online databases such as Vipul Razor or Distributed checksum clearinghouse (DCC).
- Messaging Server does not supply an Secure Socket Layer (SSL) version of `libspamass.so`, however, it is possible to build SpamAssassin to use openSSL.
- Perl 5.6 or later is required.

14.4.3.1 Where Should You Run SpamAssassin?

SpamAssassin can run on a separate system of its own, on the same system as the Messaging Server in a single system deployment, or on the same system as the MTA in a two-tier deployment. If Local Mail Transfer Protocol (LMTP) is used between the MTA and the message store, the filtering must be invoked from the MTA. It cannot be invoked from the message store. When SMTP is used between the MTA and the message store, it can be invoked from either one, and it can run on either system or a separate third system.

If you want to use a farm of servers running SpamAssassin, you would have to use a load balancer in front of them. The MTA is configured with only one address for the SpamAssassin server.

14.4.4 Deploying SpamAssassin

Perform the following steps to deploy SpamAssassin:

- **Install and configure SpamAssassin.** Refer to the SpamAssassin software documentation for installation and configuration information. See also “[14.4.7 SpamAssassin Options](#)” on page [491](#).
- **Load and configure the SpamAssassin client library.** This involves specifying the client library, `libspamass.so`, and configuration file to the MTA (you must create this file). See “[14.2.1 Loading and Configuring the Spam Filtering Software Client Library](#)” on page [465](#)
- **Specify what messages to filter for spam.** Messages can be filtered by user, domain, or channel. See “[14.2.2 Specifying the Messages to Be Filtered](#)” on page [466](#)
- **Specify what actions to take on spam messages.** Spam can be discarded, filed into a folder, tagged on the subject line, and so on. See “[14.2.3 Specifying Actions to Perform on Spam Messages](#)” on page [471](#)
- **Set miscellaneous filter configuration parameters as desired.** See [Table 14–1](#)

14.4.5 SpamAssassin Configuration Examples

This section describes some common SpamAssassin configuration examples:

- “[To File Spam to a Separate Folder](#)” on page [483](#)
- “[To Add a Header Containing SpamAssassin Score to Spam Messages](#)” on page [485](#)
- “[To Add the SpamAssassin Result String to the Subject Line](#)” on page [486](#)

Note – These examples use a number of options and keywords. Refer to “[12.12.5 Spam Filter Keywords](#)” on page [420](#) and [Table 14–1](#).

▼ To File Spam to a Separate Folder

This example tests messages arriving at the local message store and files spam into a folder called `spam`. The first three steps can be done in any order.

1 Create the SpamAssassin configuration file.

The name and location of this file is specified in [Step 2](#). A good name is `spamassassin.opt`. This file contains the following lines:

```
host=127.0.0.1
port=2000
mode=0
verdict=spam
debug=1
```

`host` and `port` specify the name of the system where `spamd` is running and the port on which `spamd` listens for incoming requests. `mode=0` specifies that a string, specified by `verdict`, is returned if the message is perceived as spam. `debug=1` turns on debugging in the SpamAssassin library. See [Table 14–3](#)

2 Add the following lines to the `option.dat` file:

```
! for Spamassassin
spamfilter1_config_file=/opt/SUNWmsgsr/config/spamassassin.opt
spamfilter1_library=/opt/SUNWmsgsr/lib/libspamass.so
spamfilter1_optional=1
spamfilter1_string_action=data:,require "fileinto"; fileinto "$U";
```

`spamfilter1_config_file` specifies the SpamAssassin configuration file.

`spamfilter1_library` specifies the SpamAssassin shared library.

`spamfilter1_optional=1` specifies that the MTA continue operation if there is a failure by `spamd`.

`spamfilter1_string_action` specifies the Sieve action to take for a spam messages.

In this example, `spamfilter1_string_action` is not necessary because the default value already is `data:,require "fileinto"; fileinto "$U";`. This line specifies that spam messages are sent to a folder. The name of the folder is the spam verdict value returned by SpamAssassin. The value returned by SpamAssassin is specified by the `verdict` option in `spamassassin.opt`. (See [Step 1](#).) In this case, the folder name is `spam`.

3 Specify the messages to be filtered.

To filter all messages coming into the local message store, change the `imta.cnf` file by adding the `destinationspamfilterXoptin` `spam` keywords on the `ims-ms` channel:

```
!
! ims-ms
ims-ms defragment subdirs 20 notices 1 7 14 21 28 backoff "pt5m" "pt10m"
"pt30m" "pt1h" "pt2h" "pt4h" maxjobs 4 pool IMS_POOL fileinto
$U+$S@$D destinationspamfilter1optin spam
ims-ms-daemon
```

- 4 Recompile the configuration and restart the server. Only the MTA needs to be restarted. You do not need to execute stop-msg.

```
# imsimta cnbuild
# imsimta restart
```

- 5 Start the spamd daemon. This is normally done with a command of the form:

```
spamd -d
```

spamd defaults to only accepting connections from the local system. If SpamAssassin and Messaging Server are running on different systems, this syntax is required:

```
spamd -d -i listen_ip_address -A allowed_hosts
```

where *listen_ip_address* is the address on which to listen and *allowed_hosts* is a list of authorized hosts or networks (using IP addresses) which can connect to this spamd instance.

Note – 0.0.0.0 can be used with -i *listen_ip_address* to have spamd listen on all addresses. Listening on all addresses is preferable because it `spamfilterX_verdict_n` avoids having to change command scripts when changing a system's IP address.

▼ To Add a Header Containing SpamAssassin Score to Spam Messages

This example adds the header `Spam-test: result string` to messages determined to be spam by SpamAssassin. An example header might be:

```
Spam-test: True ; 7.3 / 5.0
```

where `Spam-test:` is a literal and everything after that is the result string. `True` means that it is spam (`false` would be not spam). `7.3` is the SpamAssassin score. `5.0` is the threshold. This result is useful for setting up a Sieve filter that can file or discard mail above or between a certain score.

In addition, setting `USE_CHECK` to `0` returns the list of SpamAssassin tests that matched along with the verdict string. See `USE_CHECK` in [Table 14–3](#).

- 1 Specify the messages to be filtered. This is described in [Step 3 in “To File Spam to a Separate Folder” on page 483](#)

- 2 Create the SpamAssassin configuration file.

The name and location of this file is specified with `spamfilter_configX_file` (see next step). It consists of the following lines:

```
host=127.0.0.1
port=2000
mode=1
field=
```

```
debug=1
```

host and port specify the name of the system where `spamd` is running and the port on which `spamd` listens for incoming requests. mode=1 specifies that the SpamAssassin result string is returned if the message is found to be spam. field= specifies a string prefix for the SpamAssassin result string. In this example, a prefix is not desired because we are specifying it in the Sieve script. debug=1 turns on debugging in the SpamAssassin library.

3 Add the following lines to the option.dat file:

```
!for Spamassassin
spamfilter1_config_file=/opt/SUNWmsgsr/config/spamassassin.opt
spamfilter1_library=/opt/SUNWmsgsr/lib/libspamass.so
spamfilter1_optional=1
spamfilter1_string_action=data:,require ["addheader"];addheader "Spam-test: $U";
```

As in previous examples, the first three options specify the SpamAssassin configuration file, shared library, and to continue MTA operation if there is a failure by the shared library. The following line:

```
spamfilter1_string_action=data:,require ["addheader"];addheader "Spam-test: $U";
```

specifies that a header is added to spam messages. The header has the literal prefix `Spam-test:` followed by the string returned by SpamAssassin. Because mode=1 was specified in the previous step, the SpamAssassin result string is returned. For example: True; 7.3/5.0

4 Recompile the configuration, restart the server and start the `spamd` daemon.

See “[14.4.5 SpamAssassin Configuration Examples](#)” on page 483.

▼ To Add the SpamAssassin Result String to the Subject Line

By adding the SpamAssassin result string to the Subject line, users can determine whether they wish to read a message with a SpamAssassin score. For example:

Subject: [SPAM True ; 99.3 / 5.0] Free Money At Home with Prescription Xanirex!

Note that setting `USE_CHECK` to 0 returns the list of SpamAssassin tests that matched along with the verdict string (see “[14.4.7 SpamAssassin Options](#)” on page 491 in “[14.4.7 SpamAssassin Options](#)” on page 491). This list can be very long, so it is best to set `USE_CHECK` to 1.

1 Specify the messages to be filtered.

See Step 3 in “[To File Spam to a Separate Folder](#)” on page 483

2 Create the SpamAssassin configuration file.

This step is described in “[To File Spam to a Separate Folder](#)” on page 483. mode=1 specifies that the SpamAssassin result string is returned if the message is found to be spam.

```
host=127.0.0.1  
port=2000  
mode=1  
debug=1
```

host and port specify the name of the system where spamd is running and the port on which spamd listens for incoming requests. mode=1 specifies that the SpamAssassin result string is returned if the message is spam. debug=1 turns on debugging in the SpamAssassin library.

3 Add the following lines to the option.dat file:

```
!for Spamassassin  
spamfilter1_config_file=/opt/SUNWmsgsr/config/spamassassin.opt  
spamfilter1_library=/opt/SUNWmsgsr/lib/libspamass.so  
spamfilter1_optional=1  
spamfilter1_string_action=data:, addtag "[SPAM detected: $U]";
```

As in previous examples, the first three options specify the SpamAssassin configuration file, shared library, and to continue MTA operation if there is a failure by the shared library. The following line

```
spamfilter1_string_action=data:, addtag "[SPAM detected $U]";
```

specifies that a tag be added to the Subject: line. It has the literal prefix SPAM detected followed by the field string (default: Spam-Test) followed by “[result string]” returned by SpamAssassin. Because mode=1 was specified in “[14.4.5 SpamAssassin Configuration Examples](#)” on page 483, the SpamAssassin result string is returned. Thus, a subject line looks something like this:

```
Subject: [SPAM detected Spam-Test: True ; 11.3 / 5.0] Make Money!
```

You can also use addheader and addtag together:

```
spamfilter1_string_action=data:, require ["addheader"]; addtag "[SPAM detected  
$U]"; addheader "Spamscore: $U";
```

to get a message like this:

```
Subject: [SPAM detected Spam-Test: True ; 12.3 / 5.0] Vigaro Now!Spamscore:  
Spam-Test: True ; 12.3 / 5.0
```

Set field= in spamassassin.opt to remove the default value of Spam-Test. A cleaner message is returned:

```
Subject: [SPAM True ; 91.3 / 5.0] Vigaro Now!Spamscore: True ; 91.3 / 5.0
```

4 Recompile the configuration, restart the server and start the `spamd` daemon.

See “[To File Spam to a Separate Folder](#)” on page 483.

▼ **To Filter Messages Based on SpamAssassin Score**

This example shows how to filter messages based on a SpamAssassin score. It uses the `spamadjust` and `spamtest` Sieve filter actions. In this example, a header containing the SpamAssassin score is added to all messages. This header can be used by the SpamAssassin software administrator to tune SpamAssassin for improved spam email detection. If the message has a SpamAssassin score between 5 and 10, the message is filtered to a `spam` folder within the user's account. If the message has a SpamAssassin score greater than 10, the message is discarded. Note that by default SpamAssassin considers messages with a score of 5 and greater to be spam.

1 Specify the messages to be filtered.

This is described in [Step 3 of “To File Spam to a Separate Folder” on page 483](#).

2 Create the SpamAssassin configuration file.

The name and location of this file is specified with `spamfilter_configX_file` (see next step). It consists of the following lines:

```
debug=1  
host=127.0.0.1  
port=783  
mode=2  
field=
```

`host` and `port` specify the name of the system where `spamd` is running and the port on which `spamd` listens for incoming requests. `mode=2` specifies that the SpamAssassin result string is always returned regardless of the score. `field=` specifies a string prefix for the SpamAssassin result string. In this example, a prefix is not desired because we are specifying it in the Sieve script. `debug=1` turns on debugging in the SpamAssassin library.

3 Add the following lines to the option.dat file

```
! For SpamAssassin  
spamfilter1_config_file=/opt/SUNWmsgsr/config/spamassassin.opt  
spamfilter1_library=/opt/SUNWmsgsr/lib/libspamass.so  
spamfilter1_optional=1  
spamfilter1_string_action=data:, require ["addheader","spamtest"]; \  
spamadjust "$U"; addheader "Spam-test: $U"
```

As in previous examples, the first three lines specify the SpamAssassin configuration file, shared library, and to continue MTA operation if there is a failure by the shared library. The last two lines specify that the SpamAssassin score should be extracted from the return string from SpamAssassin (`$U`), which is used in the `spamtest` operation, and a spam score header should be added to all messages (for example, `Spam-test: True; 7.3/5.0`)

4 Create a channel level filter to process the email based on the spam score.

Refer to “[To Create a Channel-level Filter](#)” on page 572. Add the following rule to that file:

```
require ["spamtest","relational","comparator-i;ascii-numeric","fileinto"];
if spamtest :value "ge" :comparator "i;ascii-numeric" "10" {discard;}
elsif spamtest :value "ge" :comparator "i;ascii-numeric" "5" {fileinto "spam";}
else {keep;}
```

The second line discards the spam email if the SpamAssassin score is greater or equal to 10. The third line files the email to the users "spam" folder if the score is greater or equal to 5. The last line else {keep;} keeps all messages which received a score less than 5.

5 Recompile the configuration, restart the server and start the `spamd` daemon

See the final steps in “[To File Spam to a Separate Folder](#)” on page 483.

14.4.6 Testing SpamAssassin

To test SpamAssassin, first set debug=1 in the `spamassassin.opt` file. You do not have to turn on the channel-specific `master_debug` or `slave_debug` in the `imta.cnf`. Then send a test message to a test user. The `msg-svr-base/data/log/tcp_local_slave.log*` file should have lines similar to these:

```
15:15:45.44: SpamAssassin callout debugging enabled; config
/opt/SUNWmsgsr/config/spamassassin.opt
15:15:45.44: IP address 127.0.0.1 specified
15:15:45.44: Port 2000 selected
15:15:45.44: Mode 0 selected
15:15:45.44: Field "Spam-Test: " selected
15:15:45.44: Verdict "spam" selected
15:15:45.44: Using CHECK rather than SYMBOLS
15:15:45.44: Initializing SpamAssassin message context
...
15:15:51.42: Creating socket to connect to SpamAssassin
15:15:51.42: Binding SpamAssassin socket
15:15:51.42: Connecting to SpamAssassin
15:15:51.42: Sending SpamAssassin announcement
15:15:51.42: Sending SpamAssassin the message
15:15:51.42: Performing SpamAssassin half close
15:15:51.42: Reading SpamAssassin status
15:15:51.67: Status line: SPAMD/1.1 0 EX_OK
15:15:51.67: Reading SpamAssassin result
15:15:51.67: Result line: Spam: False ; 1.3 / 5.0
15:15:51.67: Verdict line: Spam-Test: False ; 1.3 / 5.0
15:15:51.67: Closing connection to SpamAssassin
15:15:51.73: Freeing SpamAssassin message context
```

If your log file does not contain lines similar to these, or if `spamd` is not running, the following error message is returned in your SMTP dialog after the last period (.) is sent to the SMTP server.

```
452 4.4.5 Error writing message temporaries - Temporary scan failure: End  
message status = -1
```

In addition, if `spamfilter1_optional=1` (highly recommended) is set in `option.dat`, the message is accepted, but not filtered. It is as if spam filtering was not enabled, and the following lines appear in `tcp_local_slave.log*`:

```
15:35:15.69: Creating socket to connect to SpamAssassin  
15:35:15.69: Binding SpamAssassin socket  
15:35:15.69: Connecting to SpamAssassin  
15:35:15.69: Error connecting socket: Connection refused  
15:35:15.72: Freeing SpamAssassin message context
```

The call to SpamAssassin happens after the SMTP server received the entire message (that is, after the last “.” is sent to the SMTP server), but before the SMTP server acknowledges to the sender that it accepted the message.

Another test is to send a sample spam message using `sample-spam.txt` from, for example, the `Mail-SpamAssassin-2.60` directory. This message has the following special text string in it:

```
XJS*C4JDBQADN1.NSBN3*2IDNEN*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL*C.34X
```

The corresponding `tcp_local_slave.log*` contains something like this:

```
16:00:08.15: Creating socket to connect to SpamAssassin  
16:00:08.15: Binding SpamAssassin socket  
16:00:08.15: Connecting to SpamAssassin  
16:00:08.15: Sending SpamAssassin announcement  
16:00:08.15: Sending SpamAssassin the message  
16:00:08.15: Performing SpamAssassin half close  
16:00:08.15: Reading SpamAssassin status  
16:00:08.43: Status line: SPAMD/1.1 0 EX_OK  
16:00:08.43: Reading SpamAssassin result  
16:00:08.43: Result line: Spam: True ; 1002.9 / 5.0  
16:00:08.43: Verdict line: Spam-Test: True ; 1002.9 / 5.0  
16:00:08.43: Closing connection to SpamAssassin  
16:00:08.43: Mode 0 verdict of spam  
16:00:08.43: Mode 0 verdict of spam  
16:00:08.47: Freeing SpamAssassin message context
```

A corresponding entry in the `mail.log_current` file would look as follows. Note the `+spam` part of the destination address, which means the message is filed in the folder called `spam`:

```
15-Dec-2003 15:32:17.44 tcp_intranet ims-ms E 1 morchia@siroe.com rfc822;
morchia morchia+spam@ims-ms-daemon 15-Dec-2003 15:32:18.53
ims-ms D 1 morchia@siroe.com rfc822;morchia morchia+spam@ims-ms-daemon
```

14.4.7 SpamAssassin Options

This section contains the SpamAssassin option table.

TABLE 14-3 SpamAssassin Options (spamassassin.opt)

Options	Description	Default
debug	Specifies whether to turn on debugging in the <code>libspamass.so</code> . Debugging of <code>spamd</code> itself is controlled by the command line invoking <code>spamd</code> . Set to an integer value. 0 is off, 1 is on, a setting of 2 or greater reports exactly what was received from <code>spamd</code> .	0
field	<p>Specifies the string prefix for the SpamAssassin result. SpamAssassin results look like this: “Spam-test”</p> <p><code>Spam-Test: False ; 0.0 / 5.0</code> <code>Spam-Test: True ; 27.7 / 5.0</code></p> <p>The <code>field</code> option provides the means for changing the <code>Spam-Test:</code> part of the result. Note that the “<code>:</code>” is removed if an empty <code>field</code> value is specified.</p> <p>If <code>USE_CHECK</code> is set to 0, the result string will look similar to this:</p> <p><code>Spam-test: False ; 0.3 / 4.5 ; HTML_MESSAGE,NO_REAL_NAME</code></p> <p><code>Spam-test: True ; 8.8 / 4.5 ; NIGERIAN_BODY,</code> <code>NO_REAL_NAME,PLING_PLING,RCVD_IN_SBL,SUBJ_ALL_CAPS</code></p>	
host	The name of the system where <code>spamd</code> is running.	localhost

TABLE 14–3 SpamAssassin Options (spamassassin.opt) (Continued)

Options	Description	Default
mode	<p>Controls the translation of SpamAssassin filter results to verdict information. That is, it specifies what verdict information is returned after a message is processed. Four modes are available. See “14.4.7.1 The SpamAssassin mode Option” on page 493 for further explanation.</p> <p>0 - Return a <i>verdict string</i> (specified by the <code>verdict</code> option), if the message is spam. The MTA option <code>spamfilterX_string_action</code> can be used to specify what to do if a <i>verdict</i> string is returned. If the <code>verdict</code> option (defined below) is empty or unspecified, and message is spam, a <i>null verdict</i> is returned. The MTA option <code>spamfilterX_null_action</code> can be used to specify what to do if a null verdict is returned.</p> <p>Returns a <i>SpamAssassin default result string</i> if it is not spam. (A default verdict always means to take no action and deliver as normal.)</p> <p>1 - Returns the SpamAssassin <i>result string</i> if the message is found to be spam. Returns a <i>SpamAssassin default result string</i> if it is not spam. (Again, a default verdict always means to take no action and deliver as normal.) A SpamAssassin result string looks something like this: <code>True; 6.5 / 7.3</code>.</p> <p>2 - Same as mode 1 except that the SpamAssassin result string is returned regardless of whether the message is spam or not spam. No default or null verdict is ever returned and the <code>verdict</code> option is never used.</p> <p>3 - Return the SpamAssassin result string if the message is found to be spam; return the <i>verdict</i> string specified by the <code>verdict</code> option if it is not. You can control the action for the SpamAssassin result string by using the <code>spamfilterX_verdict_n</code> and <code>spamfilterX_action_n</code> matched pair. You can control the action for the <i>verdict</i> string by using <code>spamfilterX_string_action</code>.</p>	0
port	Specifies the port number where <code>spamd</code> listens for incoming requests.	783
USE_CHECK	1 - The <code>spamd CHECK</code> command is used to return the SpamAssassin score. 0 - Enables use of the <code>SYMBOLS</code> command which returns a score and a list of the SpamAssassin tests that matched. The system may hang or have other problems with this option in pre-2.55 versions of SpamAssassin. See <i>field</i> above.	
SOCKS_HOST	String. Specifies the name of an intermediate SOCKS server. If this option is specified the ICAP connection is made through the specified SOCKS server and not directly.	""
SOCKS_PORT	Specifies the port that the intermediate SOCKS server is running on.	1080
USERNAME_MAPPING	Specify the name of a mapping to probe with address information as the plugin receives recipient addresses from the MTA. The probe format is: <code>current-username current-recipient-address current-optin-string</code> If the mapping sets the \$Y flag the output string is taken to be the updated username to pass to <code>spamd</code> .	""
verdict	Specifies the verdict string used for MODE 0.	«»

14.4.7.1 The SpamAssassin mode Option

After processing a message, SpamAssassin determines whether a message is spam or not. `mode` allows you to specify the string that is returned to indicate the verdict. The string choices are null, default, SpamAssassin result string, or a verdict string specified with the `verdict` option. (Note that `default` is neither null, the SpamAssassin result string, nor the string specified by `verdict`, but some other non-configurable result string.) The mode operations are outlined in the table below.

TABLE 14-4 Returned String for the SpamAssassin mode Option

<code>verdict\Setting</code>	<code>Spam?</code>	<code>mode=0</code>	<code>mode=1</code>	<code>mode=2</code>	<code>mode=3</code>
<code>verdict="" (not set)</code>	yes	null	SpamAssassin result	SpamAssassin result	SpamAssassin result
	no	default	default	SpamAssassin result	default
<code>verdict=string</code>	yes	<code>verdict string</code>	SpamAssassin result	SpamAssassin result	SpamAssassin result
	no	default	default	SpamAssassin result	<code>verdict string</code>

The first column indicates whether the `verdict` option is set or not set. The second column indicates whether the message is spam or not. The mode columns indicate the string returned for the various modes. For example, if `verdict` is not set and `mode` is set to 0 and a message is not spam, a default string is returned. If the `verdict` is set to YO SPAM! and `mode` is set to 0 and a message is spam, the string YO SPAM! is returned.

14.5 Using Symantec Anti-Virus Scanning Engine (SAVSE)

In addition to describing how to deploy SAVSE this section can also be useful in deploying other ICAP-supported anti-spam/anti-virus programs. This section consists of the following subsections:

- “14.5.1 SAVSE Overview” on page 494
- “14.5.2 SAVSE Requirements and Usage Considerations” on page 494
- “14.5.3 Deploying SAVSE” on page 494
- “14.5.4 SAVSE Configuration Example” on page 495
- “14.5.5 SAVSE Options” on page 497

14.5.1 SAVSE Overview

SAVSE is a TCP/IP server application and communication Application Programming Interface (API) that provides virus scanning services. Designed specifically to protect traffic served through, or stored on, network infrastructure devices, it detects and protects against viruses, worms, and Trojan horses in all major file types, including mobile code and compressed file formats. Refer to the Symantec website for detailed information

Note – The current version of Messaging Server only supports the SAVSE scan function. It does not support the repair or delete functions.

14.5.2 SAVSE Requirements and Usage Considerations

This is a separately licensed product from Symantec.

Only the scan mode is supported, not the scan and repair or scan and delete mode in the SAVSE configuration.

14.5.2.1 Where Should You Run SAVSE?

SAVSE or another server that supports ICAP can run on a separate system of its own, on the same system as the Messaging Server in a single system deployment, or in a two-tier deployment on the same system as the MTA. If Local Mail Transfer Protocol (LMTP) is used between the MTA and the message store, the filtering must be invoked from the MTA. It cannot be invoked from the message store. When SMTP is used between the MTA and the message store, it can be invoked from either one, and it can run on either system or a separate third system.

If you want to use a farm of servers running SAVSE, you would have to use a load balancer in front of them. The MTA is configured with only one address for the SAVSE server.

14.5.3 Deploying SAVSE

Perform the following steps to deploy SAVSE.

- **Install and configure the SAVSE.** Refer to the Symantec software documentation for installation and configuration information. See also “[14.5.5 SAVSE Options](#)” on page [497](#).
- **Load and configure the SAVSE client library.** This involves specifying the client library `libicap.so` and configuration file to the MTA (you must create this file). See “[14.2.1 Loading and Configuring the Spam Filtering Software Client Library](#)” on page [465](#)
- **Specify what messages to filter for viruses.** Messages can be filtered by user, domain, or channel. See “[14.2.2 Specifying the Messages to Be Filtered](#)” on page [466](#)

- **Specify what actions to take on virus messages.** Viruses can be discarded, filed into a folder, tagged on the subject line, and so on. See “[14.2.3 Specifying Actions to Perform on Spam Messages](#)” on page 471
- **Set miscellaneous filter configuration parameters as desired.** See [Table 14–1“14.2.3 Specifying Actions to Perform on Spam Messages”](#) on page 471

14.5.4 SAVSE Configuration Example

The following example tests messages arriving at the local message store and discards messages with attached viruses. The first three steps can be done in any order.

▼ To Configure SAVSE

1 Create the SAVSE configuration file.

The name and location of this file is specified in the next step. The name used here is SAVSE.opt. An example of this file is shown below:

```
host=127.0.0.1
port=1344
mode=0
verdict=virus
debug=1
```

host and port specify the name of the system where the SAVSE program is running and the port (1344 is the default for SAVSE) on which it listens for incoming requests. mode=0 specifies that a string, specified by verdict (in this case the word virus), will be returned if the message is perceived to contain a virus. debug=1 turns on debugging. See “[14.5.5 SAVSE Options](#)” on page 497 for a description of the ICAP configuration parameters.

2 Create an option.dat file. Example:

```
! for Symantex Anti-virus Scan Engine
spamfilter1_config_file=/opt/SUNWmsgsr/config/SAVSE.opt
spamfilter1_library=/opt/SUNWmsgsr/lib/libicap.so
spamfilter1_optional=1
spamfilter1_string_action=data:,discard
```

spamfilter1_config_file specifies the SAVSE configuration file.

spamfilter1_library specifies the location of the SAVSE shared library.

spamfilter1_optional=1 specifies that the MTA continue operation if there is a failure by the SAVSE program.

`spamfilter1_string_action` specifies the Sieve action to take for a spam messages. This value specifies that messages with viruses are discarded. Since this is the default value, you don't have to specify it unless you are changing the value.

3 Specify the messages to be filtered.

To filter all messages coming into the local message store, change the `imta.cnf` file by adding the `destinationspamfilterloptin` `spam` keywords on the `ims-ms` channel:

```
!
! ims-ms
ims-ms defragment subdirs 20 notices 1 7 14 21 28 backoff "pt5m" "pt10m"
"pt30m" "pt1h" "pt2h" "pt4h" maxjobs 4 pool IMS_POOL fileinto
$U+$S@$D destinationspamfilterloptin virus
ims-ms-daemon
```

4 Recompile the configuration and restart the server. Only the MTA needs to be restarted. You do not need to execute stop-msg.

```
# imsimta cnbuild
# imsimta restart
```

5 Make sure SAVSE is started.

It should have started automatically, but if not, the start command might look something like this: `/etc/init.d/symcscna start`

14.5.4.1

Other Possible Configurations

Setting `mode` to 0 can be used with a `spamfilterX_null_option` to take some other action, such as filing messages in a particular folder when they are determined to be spam. For example:

```
spamfilter1_null_option=data:,require "fileinto"; fileinto "VIRUS";
```

Note that filing infected messages into a folder is not a good idea in most cases.

Setting `mode` to 1 can be used to start an action. For example, the spam result could be included in the reject message by setting `mode` to 1 and the `spamfilterX_string_action` option in the MTA to something like:

```
spamfilter1_string_action=data:,require "reject"; reject "Message contained a
virus [$U]";
```

Like `fileinto`, using the `reject` action to deal with viruses is rarely a good idea because it sends the virus back to the sender.

You could also add a tag to the spam message header by adding a line to the `option.dat` file. Example:

```
spamfilter1_string_action=data:,addtag "[SPAM detected!]";
```

Setting mode to 2 can be used where an action needs to be taken regardless of whether or not the message was determined to contain a virus. The addition of a header field that can subsequently be tested is an obvious application for mode 2:

```
spamfilterX_string_action=data:, require ["addheader"]; addheader "$U"
```

14.5.5 SAVSE Options

The SAVSE option file is really a more generic ICAP option file. Its name and location is set by `spamfilterX_config_file` in `option.dat`. It consists of lines of the form `option=value`. The one required option is `HOST`. It must be set to the name of system where the ICAP filtering server is running. This option must be set even if the ICAP server is running on the local host. The option file is shown below.

TABLE 14-5 ICAP Options

Options	Description	Default
debug	Enables or disables debug output from the ICAP interface module. 0 or 1.	0
field	Specifies the prefix for the ICAP result. SAVSE result strings look like this: Virus-Test: False Virus-Test: True; W32.Mydoom.A@mm.enc This option provides a way to change the <code>Virus-Test:</code> part of the result. Note that the <code>:</code> is removed if an empty <code>field</code> value is specified.	Virus-test
host	The name of the system where the ICAP filtering server is running	localhost

TABLE 14-5 ICAP Options *(Continued)*

Options	Description	Default
mode	<p>Controls the translation of ICAP filter results to verdict information. That is, it specifies the string information returned after a message is processed. Four modes are available. See “14.5.1 The ICAP mode Option” on page 498 for further explanation</p> <p>0 - Returns a <i>verdict string</i> (specified by the <i>verdict</i> option), if the message contains a virus. The MTA option <i>spamfilterX_string_action</i> can be used to specify what to do if a <i>verdict</i> string is returned. If the <i>verdict</i> option is empty or not set, a <i>null verdict</i> is returned. The MTA option <i>spamfilterX_null_action</i> can be used to specify what to do if a null verdict is returned and if you want to override the default action, which is to discard the message.</p> <p>If the message does not contain a virus, a default string is returned. A default string is unconfigurable and always means to take no action and deliver as normal.</p> <p>1 - Return the ICAP <i>result string</i> if the message is found to contain a virus. If the message does not contain a virus, a default string is returned. A default string always means to take no action and deliver as normal. Below are two examples of a ICAP result string:</p> <p>VIRUS TEST: FALSEVIRUS-TEST: TRUE; W32.Mydoom.A@mm.enc</p> <p>2 - Return an ICAP result string unconditionally; no default or null verdict is ever returned and the <i>verdict</i> option is never used. This setting is intended for cases in which an action needs to be taken regardless of whether or not the message was determined to contain a virus. The addition of a header field that can subsequently be tested is an obvious application for mode 2:</p> <pre>spamfilterX_string_action=data:,require ["addheader"];addheader "\$U"</pre> <p>3 - Return the ICAP result string if the message is found to contain a virus; return the <i>verdict</i> string specified by the <i>verdict</i> option if it does not. This setting is intended for cases in which one action needs to be taken if a virus is found and another taken if one is not. You can control the action for the ICAP result string by using the <i>spamfilterX_verdict_n</i> and <i>spamfilterX_action_n</i> matched pair. You can control the action for the <i>verdict</i> string by using <i>spamfilterX_string_action</i>.</p>	0
port	Specifies the port number on which the ICAP server is running.	1344
SOCKS_HOST	String. Specifies the name of an intermediate SOCKS server. If this option is specified, the ICAP connection is made through the specified SOCKS server and not directly.	""
SOCKS_PORT	Integer. Specifies the port on which the intermediate SOCKS server is running.	1080
verdict	Specifies the verdict string used for MODE 0 and 3.	""

14.5.5.1

The ICAP mode Option

After processing a message, ICAP anti-virus programs like SASVE determines whether a message has a virus or not. *mode* allows you to specify the string returned by the ICAP program indicating this verdict. The string choices are *null*, *default*, *ICAP result string*, or a *verdict string* (specified with the *verdict* option). Note that *default* is not null, the ICAP result string,

nor the string specified by `verdict`, but some other non-configurable string returned by the program. The mode operations are outlined in the table below.

TABLE 14-6 Returned Verdict String for the ICAP mode Option

<code>verdict\Setting</code>	<code>Virus?</code>	<code>mode=0</code>	<code>mode=1</code>	<code>mode=2</code>	<code>mode=3</code>
<code>verdict="" (not set)</code>	yes	null	ICAP result	ICAP result	ICAP result
	no	default	default	ICAP result	default
<code>verdict=string</code>	yes	<code>verdict string</code>	ICAP result	ICAP result	ICAP result
	no	default	default	ICAP result	<code>verdict string</code>

The first column indicates whether the `verdict` option is set or not set. The second column indicates whether the message contains a virus or not. The mode columns indicate the string returned for the various modes. For example, if `verdict` is not set and `mode` is set to 0 and a message does not have a virus, the ICAP program returns a default. If the `verdict` is set to `WARNING VIRUS!` and `mode` is set to 0 and a message does have a virus, the ICAP program returns the string `WARNING VIRUS!`

14.6 Using ClamAV

Messaging server supports the use of the popular and freely available third-party virus scanner ClamAV for the detection of virus- and Trojan horse- infected messages. Virus signatures used by ClamAV to detect newly created viruses can be automatically updated using the `freshclam` utility provided with the ClamAV software package.

Further information on ClamAV can be found at the ClamAV website.

- “[14.6.1 ClamAV/Messaging Server Theory of Operations](#)” on page 499
- “[14.6.2 ClamAV Requirements and Usage Considerations](#)” on page 500
- “[14.6.3 Deploying ClamAV](#)” on page 500
- “[To Jettison Virus- or Trojan Horse- Infected Email Using ClamAV](#)” on page 501
- “[14.6.4 Testing ClamAV](#)” on page 502
- “[14.6.5 ClamAV Options](#)” on page 503

14.6.1 ClamAV/Messaging Server Theory of Operations

ClamAV integration in Messaging Server makes use of the `clamd` daemon that is provided as part of the ClamAV package. `clamd` is a multi-threaded process that listens on a socket for requests to process messages. After processing the message, it sends back a response and closes the connection. The client portion, `clamscan` from the ClamAV installation, is not used. This function is done by a shared library called `libclamav.so`, which is part of Messaging Server.

`libclamav.so` is loaded the same way as the Brightmail SDK is loaded.

14.6.2 ClamAV Requirements and Usage Considerations

ClamAV can run on a separate system of its own, on the same system as the Messaging Server in a single system deployment, or on the same system as the MTA in a two-tier deployment. If Local Mail Transfer Protocol (LMTP) is used between the MTA and the message store, the filtering must be invoked from the MTA. It cannot be invoked from the message store. When SMTP is used between the MTA and the message store, it can be invoked from either one.

If you want to use a farm of servers running ClamAV, use a load balancer front of them. The MTA is configured with only one address for the ClamAV server.

Other considerations.

- ClamAV is free. Go to <http://clamav.net> for software and documentation.
- ClamAV integration with the MTA can be enabled for a user, a domain, or a channel.
- The ClamAV package provides a utility to regularly update virus-signatures. The utility is called `freshclam`. Refer to the ClamAV package documentation for further information.
- The `libclamav.so` library is included by default with Messaging Server 2006Q4 and above.

14.6.3 Deploying ClamAV

Perform the following steps to deploy ClamAV:

- **Install and configure ClamAV.** Refer to the ClamAV software documentation for installation and configuration information. See also “[14.6.5 ClamAV Options](#)” on page 503.
- **Load and configure the ClamAV client library.** This involves specifying the client library, `libclamav.so` and a configuration file to the MTA (you must create this file). See “[14.2.1 Loading and Configuring the Spam Filtering Software Client Library](#)” on page 465.
- **Specify what messages to filter for spam.** Messages can be filtered by user, domain, or channel. See “[14.2.2 Specifying the Messages to Be Filtered](#)” on page 466.
- **Specify what actions to take on virus messages.** See “[14.2.3 Specifying Actions to Perform on Spam Messages](#)” on page 471.
- **Set miscellaneous filter configuration parameters as desired.** See “[14.6.5 ClamAV Options](#)” on page 503

▼ To Jettison Virus– or Trojan Horse– Infected Email Using ClamAV

The following example jettisons all messages found to contain a virus or Trojan horse detected by ClamAV. The verdict string is not used.

1 Create the ClamAV configuration file.

The name and location of this file is specified in Step 2. A good name is `clamav.opt`. This file contains the following lines:

```
# more /opt/SUNWmsgsr/config/clamav.opt
! ClamAV Settings
debug=1
host=127.0.0.1
port=3310
mode=1
```

`debug=1` turns on debugging in the ClamAV library.

`host` and `port` specify the name of the system where `clamd` is running and the port on which `clamd` listens for incoming requests.

`mode=1` specifies that the ClamAV plug-in return the ClamAV result string as the verdict when a virus infected email is detected.

2 Modify the option.dat file.

Add the following lines to the `option.dat` file:

```
! ClamAV settings
spamfilter2_config_file=/opt/SUNWmsgsr/config/clamav.opt
spamfilter2_library=/opt/SUNWmsgsr/lib/libclamav.so
spamfilter2_string_action=data:,require ["jettison"]; jettison;
```

`spamfilter2_config_file` specifies the ClamAV configuration file.

`spamfilter2_library` specifies the ClamAV shared library.

`spamfilter2_string_action` specifies the Sieve action to take for a virus infected email.

3 Specify the messages to be filtered.

To filter all messages coming into the local message store, change the `imta.cnf` file by adding the `destinationspamfilterXoptin` virus keywords on the `ims-ms` channel:

```
!
! ims-ms
ims-ms defragment subdirs 20 notices 1 7 14 21 28 backoff "pt5m" "pt10m"
"pt30m" "pt1h" "pt2h" "pt4h" maxjobs 4 pool IMS_POOL fileinto
$U+$S@$D destinationspamfilter2optin virus
ims-ms-daemon
```

4 Recompile the configuration and restart the server.

Only the MTA needs to be restarted. You do not need to execute stop-msg.

```
# imsimta cnbuild  
# imsimta restart
```

5 Start the clamd daemon.

14.6.4 Testing ClamAV

To test ClamAV, first set debug=1 in the `clamav.opt` file. (You do not have to turn on the channel-specific `master_debug` or `slave_debug` in the `imta.cnf`.) Then send a file attachment to a test user which contains the EICAR virus string (http://www.eicar.org/anti_virus_test_file.htm). This string is designed to trigger virus scanners to recognize an email as virus-infected without having an actual virus attached:

```
X50!P%@AP[4\PZX54(P^)7CC]${EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

Review the test logs. The `msg-svr-base/data/log/tcp_local_slave.log*` file should have lines similar to these:

```
10:39:00.85: ClamAV callout debugging enabled;  
config /opt/SUNWmsgsr/config/clamav.opt  
10:39:00.85: IP address 127.0.0.1 specified  
10:39:00.85: Port 3310 selected  
10:39:00.85: Mode 1 selected  
10:39:00.85: Field "Virus-Test: " selected  
10:39:00.85: Verdict "" selected  
10:39:00.85: Initializing ClamAV message context  
...  
10:39:00.85: Creating socket to connect to clamd server  
10:39:00.85: Binding clamd socket  
10:39:00.85: Connecting to clamd server  
10:39:00.85: Sending ClamAV STREAM request  
10:39:00.85: Retrieving ClamAV STREAM response  
10:39:00.85: STREAM response: PORT 2003  
10:39:00.85: Creating socket to connect to clamd server data port  
10:39:00.85: Binding clamd data socket  
10:39:00.85: Connecting to clamd server data port  
10:39:00.85: Sending ClamAV the message  
10:39:00.85: Closing ClamAV data connection  
10:39:00.85: Reading ClamAV result  
10:39:00.87: Result line: stream: Eicar-Test-Signature FOUND  
10:39:00.87: Scan result: Message is infected  
10:39:00.87: Verdict line: Virus-Test: True ; Eicar-Test-Signature  
10:39:00.87: Closing connection to ClamAV
```

```

10:39:00.87: Mode 1 verdict of Virus-Test: True ; Eicar-Test-Signature
10:39:00.87: Mode 1 verdict of Virus-Test: True ; Eicar-Test-Signature
...
10:39:00.87: Freeing ClamAV message context

```

If your log file does not contain lines similar to these, or if `clamd` is not running, the following error message is returned in your SMTP dialog after the last period (.) is sent to the SMTP server:

```

452 4.4.5 Error writing message temporaries - Error
connecting to ClamAV server

```

14.6.5 ClamAV Options

The ClamAV option file is a typical messaging server-style option file consisting of lines of the form `option=value`. The one required option is `HOST`. It must be set to the name of the system where `clamd` is running. This option must be set even if `clamd` is running on the local host.

Further additional options are available for this options file are shown below.

TABLE 14-7 ClamAV Options

Option	Description	Default
DEBUG	Enables or disables debug output from the ClamAV interface module. (Debug output from <code>clamd</code> itself is controlled by options on the <code>clamd</code> command line.) The larger the value, the more debugging output will be produced. 0 produces no output. 1 provides basic debugging. 2 adds logging of TCP traffic from <code>clamd</code> .	0
FIELD	Specifies the ClamAV result string prefix. ClamAV result strings generally look something like one of the following: Virus-Test: False Virus-Test: True ; Worm.Mydoom.I	"Virus-Test"
	The FIELD option provides the means for changing the Virus-Test part of the result. Note that the ":" will also be removed if an empty FIELD value is specified.	
MESSAGE_BUFFER_SIZE	Due to the nature of the <code>clamds can/clamd</code> interface the ClamAV plugin has to buffer the message in memory before sending to ClamAV. The size of the memory buffer is controlled by this option. It defaults to 1,048,576 characters. Messages longer than this will be truncated and not sent in their entirety to ClamAV. In order to ensure that every message is scanned fully, this value should reflect the maximum message size the MTA will accept. Reducing this value may help to speed up virus scanning times, but may let through viruses undetected.	1048576

TABLE 14-7 ClamAV Options (*Continued*)

Option	Description	Default
MODE	Controls the translation of ClamAV results to verdict information. Four different modes are available: 0 - Return the verdict string specified by the VERDICT option if the message is found to contain a virus; return a default verdict if it does not. A null verdict is returned if the VERDICT option is empty or unspecified. 1 - Return the ClamAV result as a verdict if the message is found to contain a virus; return a default verdict if not. 2 - Return a ClamAV result string as the verdict unconditionally; no default or null verdict is ever returned and the VERDICT option is never used. 3 - Return the ClamAV result as a verdict if the message is found to contain a virus; return the verdict string specified by the VERDICT option if it is not.	0
PORt	Specifies the port clamd is running on.	3310
SOCKS_HOST	Specifies the name of an intermediate SOCKS server. If this option is specified the clamd connection is made through the specified SOCKS server and not directly.	3310
SOCKS_PORT	Specifies the port the intermediate SOCKS server is running on.	1080
VERDICT	Specifies the verdict string used in modes 0 and 3.	""

14.7 Support for Sieve Extensions

In addition to the standard Sieve functions, Messaging Server provides support for a number of extensions including addheader, addtag, spamtest and spamadjust. addheader and addtag are described in “To Add a Header Containing SpamAssassin Score to Spam Messages” on page 485 and “To Add the SpamAssassin Result String to the Subject Line” on page 486.

These extensions give administrators the ability to set different threshold values as well as the ability to set up white lists that override SpamAssassin verdicts. The two can even be combined to have different thresholds depending on who sent a particular message. spamadjust is a non-standard action. spamtest is described in <ftp://ftp.isi.edu/in-notes/rfc3685.txt> (<ftp://ftp.isi.edu/in-notes/rfc3685.txt>). Note that both can be used with Symantec Brightmail as well as SpamAssassin.

The internal separator character used to delimit multiple subject line tag additions for addtaghas been changed from space to vertical bar. This makes it possible to add a tag containing spaces, as some spam filters want to do. For example, previously, addtag “[Probable Spam]” was taken to mean addtag “[Probable”and add tag “spam]”. Now it is seen as a single tag, “[Probable Spam]”. This change prevents vertical bars from being used in tags.

spamtest can be used to compare SpamAssassin scores to specific values using the Sieve [RELATIONAL] extension with the "i;ascii-numeric" comparator. The SpamAssassin score is typically a real number, but spamtest forces the score into an integer value between 0 and 10 by first rounding the score to the nearest integer. Values below 0 are forced up to 0 while values above 10 are forced down to 10. Finally, the text string maintained by Messaging Server is appended to produce the test string that the spamtest test sees. :percent is supported by spamtest (see *SIEVE Email Filtering: Spamtest and Virustest Extensions draft-ietf-sieve-spamtestbis-05*). Note that spamtest can be also be used with Brightmail.

spamadjust is used to adjust the current spam score. This action takes a single string argument which is scanned for a real numeric value. This value is used to adjust the current spam score. The entire string is also appended to the current score text string. In the example shown below, the string would be “undisclosed recipients.”

Multiple spamadjust actions are allowed; each one is added to the current score. Again, the score value always starts at 0. Signed numeric values are allowed, making it possible to lower the current score as well as raise it. There is no require clause for spamadjust; the spamtest extension should be listed instead.

For example, a possible use of spamadjust with a SpamAssassin MODE setting of 2 might be:

```
spamfilterX_string_action=data:,require ["spamtest"];spamadjust "$U";
```

A system-level Sieve filter could then modify the SpamAssassin score by checking for a particular type of header and, if found, adding 5 to the SpamAssassin score:

```
require "spamtest";
if header :contains ["to", "cc", "bcc", "resent-to", "resent-cc",
    "resent-bcc"] ["<undisclosed recipients>", "undisclosed.recipients"]
{spamadjust "+5 undisclosed recipients";}
```

Note that spamadjust can be also be used with Brightmail.

Finally, a user-level Sieve script could test the resulting value, discard messages certain to be spam, file messages likely to be spam, and allow messages from addresses in the local domain to pass through:

```
require ["spamtest", "relational",
"comparator-i;ascii-numeric", "fileinto"];
if anyof (address :matches "from" ["@siroe.com", "*@*.siroe.com"])
    {keep;}
elsif spamtest :value "ge" :comparator "i;ascii-numeric" "8"
    {discard;}
elsif spamtest :value "ge" :comparator "i;ascii-numeric" "5"
    {fileinto "spam-likely";}
else
    {keep;}
```

14.8 Using Milter

This section consists of the following subsections: “[14.8 Using Milter](#)” on page 506

- “[14.8.1 Milter Overview](#)” on page 506
- “[14.8.2 Milter/Messaging Server Theory of Operations](#)” on page 506
- “[14.8.3 Milter Requirements and Usage Considerations](#)” on page 507
- “[To Deploy Milter](#)” on page 508

14.8.1 Milter Overview

Milter is the short name for Sendmail Content Management API. It also refers to software written using this API. Milter provides a plug-in interface for third-party software to validate, modify, or block messages as they pass through the MTA. Milters can process a message's connection (IP) information, envelope protocol elements, message headers, and/or message body contents, and modify a message's recipients, headers, and body. Possible uses for filters include spam rejection, virus filtering, and content control. In general, Milter seeks to address site-wide filtering concerns in a scalable way. Originally designed for sendmail, Milters written for sendmail can now be used with Messaging Server, although some limitations may apply (see below). For more information on Milter refer to the Internet.

14.8.2 Milter/Messaging Server Theory of Operations

Milters control what action is to be performed on a message. Messaging Server controls what messages are to be acted upon by a Milter using the methods described in “[14.2.2 Specifying the Messages to Be Filtered](#)” on page 466.

In sendmail, Milter consists of support code in sendmail itself and a separate `libmilter` library. Filter authors link their filters against `libmilter` to produce a server. Sendmail is then configured to connect to these Milter servers.

Messaging Server provides a library that emulates the sendmail side of the Milter interface. This allows Milters written for sendmail to be used with Messaging Server.

A few words of caution are in order here. The Milter protocol consists of a complex mixture of text and binary elements and is not well documented. Additionally, Milter semantics are tightly coupled with sendmail's way of processing messages. In particular, Milters can and usually do access a subset of the macros defined in the sendmail configuration. Messaging Server's Milter client library attempts to provide a reasonable set of sendmail macros, but it is entirely possible for a Milter to be written that depends on a specific aspect of sendmail configuration which is not currently implemented. The net result is that an arbitrary Milter pulled off the network may or may not work with this client library. If problems develop we'll try and resolve them, but we cannot possibly guarantee success with every Milter out there.

14.8.3 Milter Requirements and Usage Considerations

The Milter server can run on a separate system of its own, on the same system as Messaging Server, in a single system deployment, or, in a two-tier deployment, on the same system as the MTA. When LMTP is used between the MTA and the message store, the filtering must be invoked from the MTA, it can not be invoked from the message store. When SMTP is used between the MTA and the message store it can be invoked from either one, and it can run on either system or a third separate system.

Messaging Server supports connecting to multiple Milter servers. If you specify a domain name that translates to multiple IP addresses, the system will try all of them in order received from the DNS until one works. Some DNS servers support randomizing the order of addresses they return, so this provides a primitive load balancing/failover facility.

14.8.3.1 Supported Milter Message Modification Actions

The Milter interface currently supports the ability to add headers (SMFIF_ADDHDRS), change or delete headers (SMFIF_CHGHDRS), and quarantine messages (SMFIF_QUARANTINE). Changing the message body (SMFIF_CHGBODY), adding recipients (SMFIF_ADDRCPT) and deleting recipients (SMFIF_DELRCPT) are not supported at the present time.

14.8.3.2 Macros Provided by the Milter Interface

The following macros are currently defined by the Milter interface:

`$j` Text placed in the `by` clause of `Received:` header fields. In Messaging Server this is controlled by the `RECEIVED_DOMAIN` MTA option. If the option is not set, the official host on the `local` channel is used instead.

`${client_addr}` The IP address of the SMTP client, expressed as a dotted quad value. Only set when SMTP over TCP is being used.

`$i` Queue ID for the current message. Messaging Server generates a unique ID for each session; this ID is what appears in the `$i` macro.

`${mail_addr}` The MAIL FROM address for the current transaction.

`${mail_host}` The host part of the MAIL FROM address for the current transaction.

`${rcpt_addr}` A RCPT TO address for the current transaction.

`${rcpt_host}` The host part of the current RCPT TO address.

▼ To Deploy Milter

Perform the following steps to deploy Milter.

- 1 Obtain and configure a Milter that will perform the actions you desire.**
Refer to specific Milter documentation for obtaining and configuring information.
- 2 Load and configure the Milter client library. (See “[14.2.1 Loading and Configuring the Spam Filtering Software Client Library](#)” on page [465](#).)**
 - a. Specify the path to the client library, `libmilter.so`. Specify the path and name of the Milter configuration file.**

Example:

```
spamfilter1_library=/opt/SUNWmsgsr/lib/libmilter.so  
spamfilter1_config_file=/opt/SUNWmsgsr/lib/milter.opt
```

- b. Create a Milter configuration file. with the desired options.**

The Milter option file consists of lines of the form `option=value`. The two required options are `HOST` and `PORT`. `HOST` must be set to the name of the system where the Milter server is running while `PORT` must be set to the port the Milter server is configured to listen on. Note that only TCP/IP connections are supported; UNIX domain sockets cannot be specified or used.

Several additional options are also available in this options file:

`DEBUG` (integer, default 0) — Enables or disables debug output from the Milter client library. The larger the value, the more debugging output will be produced. 0 produces no output. 1 provides basic debugging. 2 adds logging of TCP traffic. (Debug output from the Milter server is typically controlled by a setting on the command line used to start the server. Note that most Milters seem to only provide the ability to direct debug output to `syslog`.)

`TIMEOUT` (integer, default 3600) — Specifies the timeout in hundredths of seconds for operations involving the Milter connection. Available in 6.3 and later versions.

`SOCKS_HOST` (string, default "") — Specifies the name of an intermediate SOCKS server. If this option is specified the Milter connection is made through the specified SOCKS server and not directly.

`SOCKS_PORT` (integer, default 1080) — Specifies the port the intermediate SOCKS server is running on.

`SOCKS_PASSWORD` (string, default "") — Specifies a password to use in establishing the connection through the SOCKS server. Whether or not a username/password is required depends on the SOCKS server configuration.

3 Specify what messages to send to Milter.

Messages can be filtered by user, domain, or channel. See “[14.2.2 Specifying the Messages to Be Filtered](#)” on page 466.

4 Set the `spamfilterX_string_action` option in the `option.dat` file:

```
spamfilterX_string_action=data:, $M
```

This setting is used unconditionally, but must be in the MTA options file for Milter to work properly.

14.9 Cloudmark Anti-Abuse Client

Messaging Server also works with the Cloudmark Anti-Abuse Client. See [Configuring Cloudmark with Sun Java System Messaging Server \(<http://wikis.sun.com/display/CommSuite/Configuring+Cloudmark+with+Sun+Java+System+Messaging+Server>\)](#).

14.10 Other Anti-Spam and Denial-of-Service Technologies

Adding spam and virus filtering software to the system is the most effective way of reducing spam and viri from getting into user's mailboxes. However, Messaging Server provides a number of other techniques and methods to support spam filtering. These technologies are often used for purposes other than just spam filtering and so are spread throughout this book. Listed below are some sections describing anti-spam and denial-of-service technologies.

Anti-Spam Technologies:

- “[14.10.1 Anti-Spam Technique: Delay Sending the SMTP Banner](#)” on page 510
- “[12.12.6 Routing After Address Validation But Before Expansion](#)” on page 421
- Chapter 15, “[Handling Forged Email Using the Sender Policy Framework](#)”
- Chapter 18, “[Mail Filtering and Access Control](#)”
- “[23.7 Configuring Client Access to POP, IMAP, and HTTP Services](#)” on page 737
- “[12.4.2.6 DNS Domain Verification](#)” on page 366
- “[12.5.9 Expansion of Multiple Addresses](#)” on page 388
- “[18.14 To Create MTA-Wide Filters](#)” on page 573
- “[18.7 Configuring SMTP Relay Blocking](#)” on page 560

Denial of Service Technologies:

- Chapter 19, “[Throttling Incoming Connections Using MeterMaid](#)”
- “[12.9.2 Specifying Absolute Message Size Limits](#)” on page 411
- “[18.3.6 To Limit Specified IP Address Connections to the MTA](#)” on page 555
- “[27.4.1 Monitoring the Size of the Message Queues](#)” on page 872
- “[27.4.3 Monitoring Inbound SMTP Connections](#)” on page 873

14.10.1 Anti-Spam Technique: Delay Sending the SMTP Banner

A useful spam-fighting strategy is to delay sending the SMTP banner for a brief time (half a second, say), then clear the input buffer, and finally send the banner. The reason this works is that many spam clients are not standards-compliant and starts spewing SMTP commands as soon as the connection is open, ignoring any responses the server sends. Spam clients that do this when this capability is enabled will lose the first few commands in the SMTP dialogue, rendering the remainder of the dialogue invalid.

This feature has now been implemented in Messaging Server. It can be enabled unconditionally by setting the `BANNER_PURGE_DELAY` SMTP channel option to the number of centiseconds to delay before purging and sending the banner. A value of 0 disabled both the delay and purge.

The `PORT_ACCESS` mapping can also be used to control this capability. Specifying `$D` in the template causes an additional argument to be read from the template result after the mandatory SMTP auth rulset and realm, and optional application information addition. This value must be an integer with the same semantics as the `BANNER_PURGE_DELAY` value. Note that any `PORT_ACCESS` mapping setting overrides the `BANNER_PURGE_DELAY` SMTP channel option.

Handling Forged Email Using the Sender Policy Framework

Spam producers and email scammers often forge email by using false domain names and email addresses or by using legitimate domain names and email addresses in order to fool users into thinking that a message is from someone or some company they know. For example, a spammer could send email from an address such as president@whitehouse.gov and the user could be fooled into thinking the mail was actually from this address. Forging email may fool users into opening the unsolicited message, or worse, provide information to a false authority. Also, spammers prefer to send their email from legitimate domains that are not on an RBL list.

Sender Policy Framework (SPF) is a technology that can detect and reject forged email during the SMTP dialogue. Specifically, SPF is a protocol that allows a domain to explicitly authorize the hosts that may use its domain name. In addition, a receiving host may be configured to check this authorization. SPF can thus significantly reduce the instances of forged email.

- “15.1 Theory of Operations” on page 511
- “15.2 Limitations” on page 513
- “15.3 Pre-Deployment Considerations” on page 514
- “15.4 Setting up the Technology” on page 514
- “15.5 Reference Information” on page 514
- “15.6 Testing SPF using spfquery” on page 516
- “15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)” on page 518

15.1 Theory of Operations

When a message comes into Messaging Server, the MTA does an SPF query to determine if the address actually came from the domain on the address. An SPF query consults the DNS for TXT records belonging to the domain of the message (*domain*). *Domain* is either the domain name specified as the argument for HELO or EHLO (if the `spfhe1o` channel keyword is used) or the domain name in the originator's address given in the MAIL FROM: command (typically the part after the @ character). If no domain name is specified or available, the one specified during

HELO/EHLO is used as *domain*. Note that most ISPs distribute an authorized list of IP addresses that match their domains. If the IP address does not match the domain name, then the message is assumed to be forged.

Note – Prior to querying the DNS, we check the SPF_LOCAL mapping table for a match for domain. If a match is found there, it will be used first.

If a record found from the mapping table contains a `redirect=domain` clause, then the redirection to domain will be done as a DNS query, skipping the recursive and redundant mapping file check.

An example of a resulting TXT record is:

```
v=spf1 +mx a:colo.siroe.com/28 -all
```

The `v=spf1` token is required for SPF records supported by this RFC.

`+mx` directs us to check MX records for *domain* and confirm that the source IP address for this SMTP connection matches one of the IP addresses given as a result of an MX query for *domain*. If there is a match, the `+` means that the result of this is Pass.

`a:colo.siroe.com/28` directs us to check for A records for `colo.siroe.com`, and then confirm that the source IP address for this SMTP connection is in the same specified CIDR subnet as the A records, comparing only 28 bits (masked against 255.255.255.240). No qualifier character was specified, so it defaults to `+` meaning that a match results in a Pass.

Finally, `-all` matches everything else and results in Fail. For a more complete description of SPF records, please refer to RFC 4408 at <http://www.ietf.org/rfc/rfc4408.txt>

SPF processing can have one of several results. The table below shows the results and their descriptions.

TABLE 15–1 SPF Processing Results

Result	Description
Pass	The lookup has passed meaning that an SPF record was found and the record validated the originating system as being authorized to use <i>domain</i> .
Fail	The lookup found a matching SPF record, however, the record explicitly denied authorization for the SMTP client to use <i>domain</i> during the SMTP transaction. The default behavior of our SPF implementation is to reject the SMTP command with a 5xx reply.

TABLE 15-1 SPF Processing Results *(Continued)*

Result	Description
SoftFail	The lookup found a matching SPF record, and the record also denies authorization for the SMTP client to use <i>domain</i> , however the denial is less strict and the record does not direct an outright failure. The default behavior of our implementation is to accept the message, but note the <code>SoftFail</code> in the Received-SPF: header for subsequent evaluation such as during Sieve processing.
Neutral	The SPF record makes no claim to the SMTP client's authorization to use <i>domain</i> . The message will be accepted. The specification requires that <code>Neutral</code> be treated the same as <code>None</code> below.
None	No matching SPF record was found, therefore no SPF processing was done.
PermError	A permanent error was encountered during SPF processing, such as syntax errors in the SPF record, DNS failures while processing nested SPF records (due to <code>include:</code> mechanism or a <code>redirect=</code> modifier), or exceeding configured limits for SPF processing while processing nested SPF records. The default behavior is to reject the SMTP command with a 5xx reply.
TempError	A temporary error was encountered during SPF processing, most likely due to DNS timeouts querying SPF records. The default behavior is to reject the SMTP command with a 4xx reply.

After SPF processing has completed, a Received-SPF: header will be written to the message documenting the result of the SPF processing. This header can then be queried during Sieve processing for subsequent consideration. Extensive debugging is available if the MTA option MM_DEBUG is enabled (>0) in the option.dat file.

15.2 Limitations

SPF is only one tool to use to fight spam, and will not address all issues. It is fairly easy for a spammer to create a domain and add an SPF TXT record that will make the domain seem legitimate. On the other hand, SPF is pretty effective for detecting forged email from established ISPs, although many TXT records will allow the SPF to not fail.

15.3 Pre-Deployment Considerations

It is important to have a very fast DNS server on your system because a DNS query for every message will be required.

15.4 Setting up the Technology

There are two steps for setting up SPF technology:

- Place channel keywords on the incoming TCP channel (typically the `tcp_local` channel, although there may be other channels if you allow channel switching from `tcp_local` to another channel). See [Table 15–2](#)
- Set up the options in the `option.dat` file. See [Table 15–3](#)

15.5 Reference Information

This section provides reference information for the SPF channel keywords and the SPF MTA options. SPF support is implemented through four channel keywords applied to the incoming `tcp_*` channel (typically `tcp_local`). The following table shows the keywords and their descriptions.

TABLE 15–2 SPF Keywords

Keyword	Description
<code>spfnone</code>	Disables SPF processing
<code>spfhe1o</code>	Enables SPF processing for the domain name specified as an argument to HELO or EHLO.
<code>spfmailfrom</code>	Enables SPF processing for the domain name provided for the originator envelope address after receiving the MAIL FROM:.
<code>spfrcptto</code>	Enables SPF process for the domain name provided for the originator envelope address after receiving the RCPT TO:. Processing is the same as <code>spfmailfrom</code> except that it is delayed in the SMTP transaction until after the RCPT TO: command has been issued and the recipient has otherwise been confirmed to be a valid recipient.

Note – `spfmailfrom` and `spfrcptto` are conflicting keywords and you should only specify one of these two keywords on the channel. You can, however, use `spfhe1o` in conjunction with either `spfmailfrom` or `spfrcptto` to perform both kinds of SPF checks.

There is additional support to establish limits on SPF processing and to control whether SMTP commands will be accepted, failed with a 4xx response (temporary failure), or failed with a 5xx response (permanent failure) for the various SPF results including: `Fail`, `SoftFail`, `PermError`, and `TempError`.

The following MTA options, in `option.dat`, can be used to place limits on SPF processing.

TABLE 15-3 SPF Limiting Options

Option	Description
<code>SPF_MAX_RECURSION</code>	Specifies the number of recursions that will be allowed into nested SPF records due to <code>include:</code> or <code>redirect=</code> . Exceeding this limit will result in a <code>PermError</code> . Default: 10 (mandated by the RFC)
<code>SPF_MAX_DNS_QUERIES</code>	Specifies the number of mechanisms or modifiers that require DNS lookups (including <code>include:</code> , <code>a:</code> , <code>mx:</code> , <code>ptr:</code> , <code>exists:</code> , <code>redirect=</code> , and <code>exp=</code>). Note that the limit is not counted as the number of actual DNS lookups, so one mechanism could lead to several DNS queries. Exceeding this limit will result in a <code>PermError</code> . Default: 10 (mandated by the RFC)
<code>SPF_MAX_TIME</code>	Specifies the number of seconds that will be allowed for the SPF processing to complete. Exceeding this value will result in a <code>TempError</code> . The default value is more generous than the RFC suggests. Default: 45

Additionally, the following MTA options in `option.dat` can be configured to control the behavior of the SMTP server in response to SPF results of `Fail`, `SoftFail`, `PermError`, and `TempError`. For each of these results, the SMTP server can send back a 2xx (success) response, 4xx (temporary failure), or 5xx (permanent failure). Also, for `Fail` and `SoftFail`, the MTA can distinguish between an SPF result as the result of an "all" mechanism versus an otherwise explicitly referenced match. You can then make a distinction between a particular result and the SPF record's default result. The valid values for any of these options is 2, 4, or 5. The values of 2, 4, or 5 correspond to 2xx, 4xx, or 5xx responses from the SMTP server as a result of getting that particular SPF status. So, for example, if `SPF_SMTP_STATUS_FAIL=2` and the SPF record explicitly blocks us with a "`-a:192.168.1.44`" (our IP address), then instead of responding with a 5xx response, we'll accept the address with a "250 OK" instead.

TABLE 15-4 SPF Failure and Error Options

Option	Description
SPF_SMTP_STATUS_FAIL	Used when the match of an SPF record is a "-" flagged mechanism other than "-all" Default: 5
SPF_SMTP_STATUS_FAIL_ALL	Used when the matching mechanism is "-all" Default: 5
SPF_SMTP_STATUS_SOFTFAIL	Used when the match of an SPF record is a "~" flagged mechanism other than "~all" Default: 2
SPF_SMTP_STATUS_SOFTFAIL_ALL	Used when the matching mechanism is "~all" Default: 2
SPF_SMTP_STATUS_TEMPERROR	Used when there is a temporary failure, usually related to DNS processing problems. Default: 4
SPF_SMTP_STATUS_PERMERROR	Used when there is a permanent failure, usually due to syntax or other technical errors found during SPF processing. (Note that this will be due to a non-local error.) Default: 5

15.6 Testing SPF using spfquery

This testing utility can be used to test SPF processing.

Note – spfquery does not test your SPF configuration. It tests what would be returned if you were to enable SPF processing.

Requirements: Must be run as a user who has access to run the Messaging Server binaries and access its libraries such as root or `mailsrv`, for example.

Location: `msg-srv-base/sbin/`

15.6.1 Syntax

```
spfquery [-i ip-address] [-s sender-email] [-h helo-domain]
[-e none | neutral | pass | fail | temperror | permerror] [-v] [-V] [?] domain
```

The following table shows the `spfquery` options and their descriptions.

TABLE 15-5 `spfquery` options

Option	Description
<code>-i ip address</code>	Specifies the IP address to be used as the remote address for the SPF query. Default is <code>127.0.0.1</code> . This option can also be <code>--ip-address</code> .
<code>-s domain</code>	The email address that will be used as if it were specified as <code>MAIL FROM: .</code> Default: <code>postmaster@domain</code> . This option can also be <code>--sender</code>
<code>-h helo-domain</code>	The domain name as if it were specified for the HELO domain. Note that this domain is not verified itself, but instead provided as supplemental information for macro processing. Default value is the same as the value you specified for <code>domain</code> . This option can also be <code>--helo-domain</code>
<code>-e result</code>	<code>spfquery</code> will compare the result of the SPF processing with what is expected and if the result is different, a message will be printed and <code>spfquery</code> will exit with a non-zero return status; result can be one of: <code>none</code> , <code>neutral</code> , <code>pass</code> , <code>fail</code> , <code>softfail</code> , <code>temperror</code> , or <code>permerror</code> . This option can also be <code>--expect</code> .
<code>-v</code>	Enables verbose output during SPF processing. This option can also be <code>--verbose</code> .
<code>-V</code>	Prints the current version of the SPF library. This option can also be <code>--version</code> .
<code>-?</code>	Prints this usage information. This option can also be <code>--help</code> .

15.6.2 Example with Debugging Enabled

```
# /opt/SUNWmsgsr/sbin/spfquery -v -i 192.168.1.3 11.spf1-test.siroe.com
Running SPF query with:
    IP address: 192.168.1.3
    Domain: 11.spf1-test.siroe.com
    Sender: postmaster@11.spf1-test.siroe.com (local-part: postmaster)
    HELO Domain: 11.spf1-test.siroe.com

15:30:04.33: -----
15:30:04.33: SPFcheck_host called:
15:30:04.33:     source ip = 192.168.1.3
15:30:04.33:     domain = 11.spf1-test.siroe.com
15:30:04.33:     sender = postmaster@11.spf1-test.siroe.com
15:30:04.33:     local_part = postmaster
15:30:04.33:     helo_domain = 11.spf1-test.siroe.com
15:30:04.33:
15:30:04.33:     Looking up "v=spf1" records for 11.spf1-test.siroe.com
15:30:04.35:     DNS query status: Pass
15:30:04.35:     "v=spf1 mx:spf1-test.siroe.com"           -all"
15:30:04.35:
15:30:04.35:     Parsing mechanism: " mx : spf1-test.siroe.com"
```

```
15:30:04.35: Assuming a Pass prefix
15:30:04.35: Processing macros in spf1-test.siroe.com
15:30:04.35: Comparing against 192.168.1.3
15:30:04.35: Looking for MX records for spf1-test.siroe.com
15:30:04.41:   mx02.spf1-test.siroe.com:
15:30:04.41:     192.0.2.22 - No match
15:30:04.41:     192.0.2.21 - No match
15:30:04.41:     192.0.2.20 - No match
15:30:04.41:     192.0.2.23 - No match
15:30:04.41:   mx01.spf1-test.siroe.com:
15:30:04.42:     192.0.2.13 - No match
15:30:04.42:     192.0.2.11 - No match
15:30:04.42:     192.0.2.12 - No match
15:30:04.42:     192.0.2.10 - No match
15:30:04.42:   mx03.spf1-test.siroe.com:
15:30:04.42:     192.0.2.32 - No match
15:30:04.42:     192.0.2.30 - No match
15:30:04.42:     192.0.2.31 - No match
15:30:04.42:     192.168.1.3 - Matched
15:30:04.42: Mechanism matched; returning Pass
15:30:04.42:
15:30:04.42: Parsing mechanism: "- all : " (not evaluated)
15:30:04.42:
15:30:04.42: SPFcheck_host is returning Pass
15:30:04.42: -----
```

15.7 Handling Forwarded Mail in SPF Using the Sender Rewriting Scheme (SRS)

As described above, SPF is a mechanism that attempts to prevent email forgery by looking up special TXT records associated with the domain in the mail FROM: (envelope from) address. This operation, which can actually involve several DNS lookups, eventually produces a list of IP addresses that are authorized to send mail from the domain. The IP address of the SMTP client is checked against this list and if it isn't found, the message may be considered to be fraudulent. Support for SPF was implemented in version 6.3 of the Messaging Server.

SPF presents serious problems for sites that provide mail forwarding services such as universities (for their alumni) or professional organizations (for their members). A forwarder ends up sending out mail from essentially arbitrary senders, which can include senders who have implemented SPF policies and which, of course, don't list the IP addresses of the forwarding system or systems as being permitted to use addresses from their domain.

The Sender Rewriting Scheme, or SRS, provides a solution to this problem. SRS works by encapsulating the original sender's address inside a new address using the forwarder's own domain. Only the forwarder's own domain is exposed for purposes of SPF checks. When the

address is used it routes the mail (usually a notification) to the forwarder, which removes the address encapsulation and sends the message on to the real destination.

Of course address encapsulation isn't exactly new. Source routes were defined in RFC 822 and provide exactly this sort of functionality, as does percent hack routing and bang paths. However, these mechanisms are all problematic on today's Internet since allowing their use effectively turns your system into an open relay.

SRS deals with this problem by adding a keyed hash and a timestamp to the encapsulation format. The address is only valid for some period of time, after which it cannot be used. The hash prevents modification of either the timestamp or the encapsulated address.

SRS also provides a mechanism for handling multi-hop forwarding without undue growth in address length. For this to work certain aspects of SRS address formatting have to be done in the same way across all systems implementing SRS.

SRS support has now been implemented for our 6.3P1 release. The following MTA options have been added:

- **SRS_DOMAIN**. This must be set to the domain to use in SRS addresses. Email sent to this domain must always be routed to a system capable SRS operations for the domain. SRS processing is handled as an overlay on top of normal address processing so nothing prevents a site from using their primary domain as the SRS domain.
- **SRS_SECRETS**. This is a comma separated list of secret keys used to encode and decode SRS addresses. The first key on the list is used unconditionally for encoding. For decoding, each key is tried in order to generate a different hash value. The decoding operation proceeds if any of the hashes match.

The ability to use multiple keys makes it possible to change secrets without service disruption: Add a second key, wait for all previously issued addresses to time out, and then remove the first key.

- **SRS_MAXAGE**. Optionally specifies the number of days before a message times out. The default if the option isn't specified is 14 days.

Every system that handles email for the selected SRS domain must be configured for SRS processing and must have all three SRS options set identically.

Setting these options is sufficient to enable SRS address decoding. Encoding is another matter - it should only be done to envelope `From:` addresses you know are associated with forwarding activity. SRS encoding is controlled by six new channel keywords: `addresssrs`, `noaddresssrs`, `destinationsrs`, `nodestinationsrs`, `sourcesrs`, and `nosourcesrs`.

Three conditions have to be met for SRS encoding to occur:

- (1) The current source channel has to be marked with `sourcesrs`. (`nosourcesrs` is the default).
- (2) The current destination channel has to be marked with `destinationsrs` (`nodestinationsrs` is the default).

- (3) The current address, when rewritten, has to match a channel marked `addresssrs` (`noaddress` is the default).

Encoding only occurs when all of these conditions are true. About the simplest setup is a pure forwarding one where all messages enter and exit on the `tcp_local` channel and all non-local addresses need SRS handling. In such a setup, `tcp_local` would be marked with the three keywords `sourcesrs`, `destinationsrs`, and `addresssrs`.

Finally, `imsimta test -rewrite` has been enhanced to show SRS encoding and decoding results for whatever address is input. For example, the address `foo@example.com` might produce the output similar to:

```
SRS encoding = SRS0=dnG=IS=example.com=foo@example.org
```

If this encoded address is rewritten it will produce the output:

```
SRS decoding = foo@example.com
```

`imsimta test -rewrite` will also show any errors that occur during SRS decoding.

LMTP Delivery

The Sun Java System Messaging Server MTA can use LMTP (Local Mail Transfer Protocol, defined in RFC 2033) for delivery to the message store in situations where a multi-tier messaging server deployment is used. In these scenarios, where you are using inbound relays and back end message stores, the relays become responsible for address expansion and delivery methods such as autoreply and forwarding and also for mailing list expansion. Delivery to the back end stores historically has been over SMTP which requires the back end system to look up the recipient addresses in the LDAP directory again, thereby engaging the full machinery of the MTA. For speed and efficiency, the MTA can use LMTP rather than SMTP to deliver messages to the back end store. The Sun Java System Messaging Server's LMTP server is not intended as a general purpose LMTP server, but rather as a private protocol between the relays and the back end message stores. For simplicity of discussion, examples involving two-tier deployments are used.

Note – By design, LMTP is intended for use in multi-tier deployments. It is not possible to use LMTP with single-system deployments. Also, the Messaging Server's LMTP service as implemented is not designed to work with other LMTP servers or other LMTP clients.

This chapter consists of the following sections:

- “[16.1 LMTP Delivery Features](#)” on page 522
- “[16.2 Messaging Processing in a Two-Tier Deployment Without LMTP](#)” on page 522
- “[16.3 Messaging Processing in a Two-Tier Deployment With LMTP](#)” on page 524
- “[16.4 LMTP Overview](#)” on page 525
- “[16.5 To Configure LMTP Delivery](#)” on page 525
- “[16.6 LMTP Protocol as Implemented](#)” on page 530

16.1 LMTP Delivery Features

The MTA's LMTP server is more efficient for delivering to the back end message store because it:

- Reduces the load on the back end stores.

Because relays are horizontally scalable and back end stores are not, it is good practice to push as much processing to the relays as possible

- Reduces the load on the LDAP servers.

The LDAP infrastructure is often a limiting factor in large messaging deployments.

- Reduces the number of message queues.

Having queues on both the relay and the back end store makes finding a lost message that much harder for the administrator of a messaging deployment.

16.2 Messaging Processing in a Two-Tier Deployment Without LMTP

[Figure 16–1](#) presents in pictorial form the following discussion of message processing in a two-tier deployment scenario without LMTP.

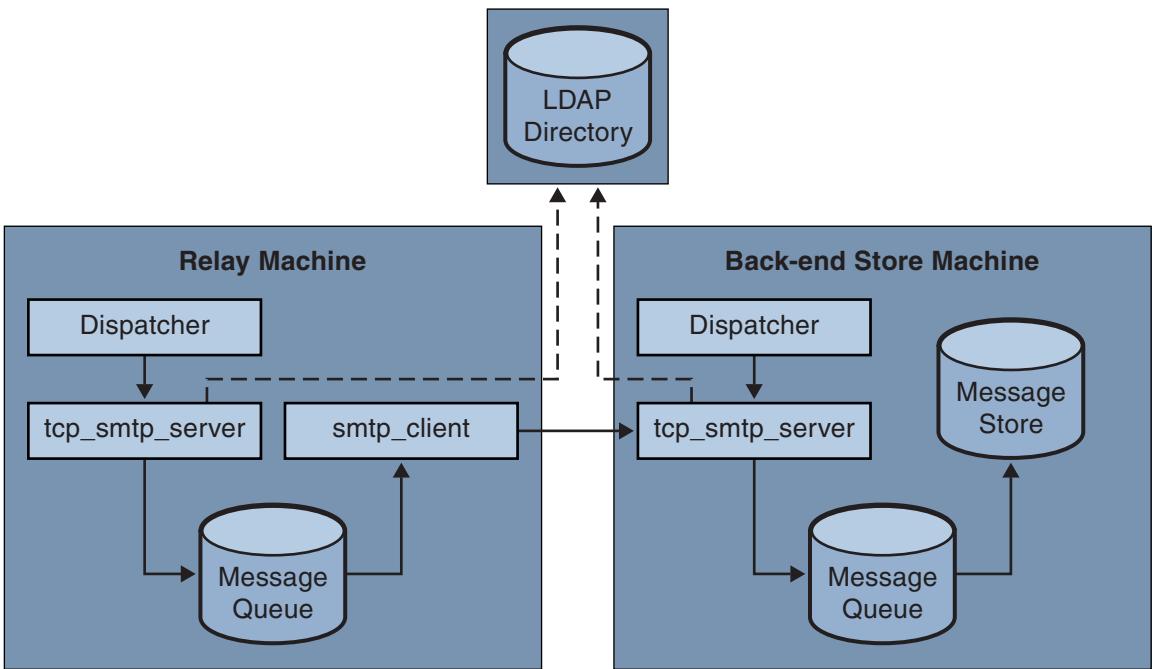


FIGURE 16-1 Two Tier Deployment Without LMTP

Without LMTP, in a two level deployment with relays in front of the store systems, the processing of an inbound message begins with a connection on the SMTP port picked up by the dispatcher on the relay machine and handed off to a `tcp_smtp_server` process. This process does a number of things with the inbound message including:

- Looking up the user in the directory
- Determining if the user is within a domain hosted by this email deployment
- Determining if the user is a valid user in the domain
- Rewriting the envelope address as `@mailhost:user@domain`
- Enqueuing the message for delivery to the mailhost

The `smtp_client` process then picks up the mail message from the queue and sends it to the mailhost. On the mailhost, some very similar processing takes place. A connection on the SMTP port is picked up by the dispatcher and handed off to a `tcp_smtp_server` process. This process does a number of things to the message, including:

- Looking up the user in the directory
- Determining if the user is within a domain hosted by this email deployment
- Determining if the user is a valid user in the domain
- Rewriting the envelope address to direct the message to the `ims_ms` channel
- Enqueuing the message for delivery to the store

Then the `ims_ms` process picks up the mail message and attempts to deliver it to the store. In this scenario, the enqueueing processing is performed twice, and the MTAs each perform an LDAP lookup.

16.3 Messaging Processing in a Two-Tier Deployment With LMTP

[“16.3 Messaging Processing in a Two-Tier Deployment With LMTP” on page 524](#) presents in pictorial form the following discussion of message processing in a two-tier deployment scenario with LMTP.

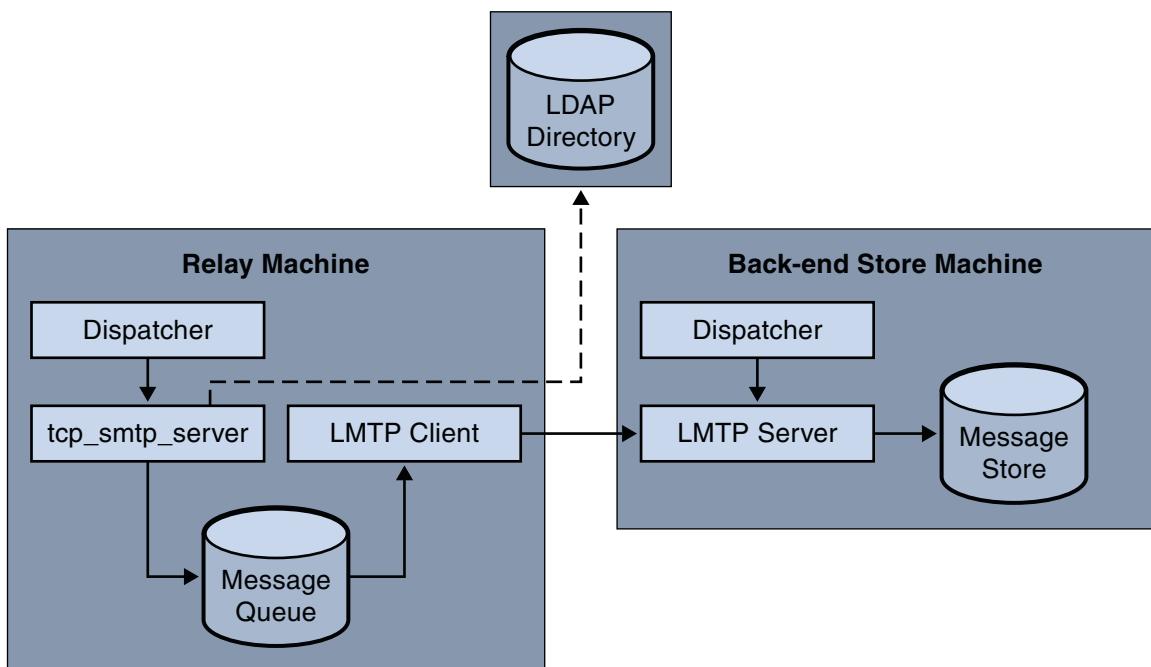


FIGURE 16–2 Two Tier Deployment With LMTP

With LMTP in place, a connection on the SMTP port of the relay machine is picked up by the dispatcher and handed off to a `tcp_smtp_server` process. This process does a number of things with the inbound message including:

- Looking up the user in the directory
- Determining if the user is within a domain hosted by this email deployment
- Determining if the user is a valid user in the domain

- Determining which back end message store machine hosts the mailbox for the user
- Enqueuing the message for delivery to the mailhost

On the store machine, a connection to the LMTP port is received by the dispatcher and handed off to the `lsmtp_server` process. The LMTP server then inserts the message into the user's mailbox or into the UNIX native mailbox. If message delivery is successful, the message is dequeued on the relay machine. If unsuccessful, the message remains on the relay machine. Note that the LMTP process on the message store does not engage any MTA machinery for processing addresses or messages.

16.4 LMTP Overview

For the most part, the MTA itself can be basically absent from the back end server. The only necessary MTA components are:

- The dispatcher
- `libimta`
- The LMTP server
- The `imta.cnf` file
- The `mappings` file
- The `imta.tailor` file

While the dispatcher requires MTA configuration files, these files can be very short. The dispatcher must run on the back end server so that it can start the LMTP servers which run under it. Because the dispatcher and the LMTP server use various functions of `libimta`, this needs to be present on the back end server as well.

The LMTP server does not perform any of the usual MTA enqueueing or dequeuing functions, header processing, or address translations. The relay system performs all the manipulation of the content of the messages and addresses which then presents to the LMTP server the message in exactly the form to be delivered to the message store and with the delivery address already in the form required by the store. Additional recipient information that is usually available as a message is delivered to the store, such as the user's quota, is presented along with the recipient address as LMTP parameters. Should a delivery attempt fail, the message is left enqueued in the LMTP queue on the relay system.

16.5 To Configure LMTP Delivery

Configuring the LMTP delivery mechanism requires configuration on both the relay machines and on the back end stores. On the relays, the `DELIVERY_OPTIONS` MTA option (in `option.dat`) has to be changed so that messages being delivered to the stores are passed to the LMTP channel. The back end store must be configured with the dispatcher, but does not need the job controller. The dispatcher must be configured to run the LMTP server.

In a typical multi-tier deployment, users are provisioned on different backend message store machines. One or more of these backend machines may not have LMTP turned on and therefore the front-end relays need to be aware of which store machines are LMTP aware. This is achieved by using the General Database facility to explicitly name those message stores which are configured to accept LMTP delivery.

▼ To Configure the Inbound MTA Relays with LMTP

To configure inbound MTA relays to use LMTP, do the following:

- 1 **Modify your imta.cnf file and change the LMTP rewrite rules to read:**

```
! lmtp
.lmtp    $E$F$U%$H.lmtp@lmtpcs-daemon
.lmtp    $B$F$U%$H@$H@lmtpcs-daemon
!
```

- 2 **Set the mailbox DELIVERY_OPTIONS to:**

```
##mailbox=@$X.LMTP:$M%$\$2I$_+$2S@lmtpcs-daemon
```

- 3 **Add the channel keywords multigate connect canonical to each of the tcp_lmtp* channel blocks.**

- 4 **Add the following channel keywords to the tcp_lmtpcs channel:**

```
fileinto @$40:$U+$S@$D
```

Note that the 'O' in the keyword above is a capital letter O, not a zero.

- 5 **The incoming MTA relay configuration settings should look like this:**

The option.dat entry for DELIVERY_OPTIONS should look like this:

```
!-----
! Modified DELIVERY_OPTIONS to activate LMTP
! delivery from a frontend to the backend store
!-----
!
DELIVERY_OPTIONS=\
  ##mailbox=@$X.LMTP:$M%$\$2I$_+$2S@lmtpcs-daemon, \
  #&members=*, \
  #&@members_offline=*, \
  #/hold=@hold-daemon:$A, \
  #program=$M%$P@pipe-daemon, \
  #forward=**, \
  #*^!autoreply=$M+$D@bitbucket
!
```

After your changes the modified `imta.cnf` rewrite rules should look like this:

```
! lmtp
.lmtp    $E$F$U$$H.lmtp@lmtpcs-daemon
.lmtp    $B$F$U%$H@$H@lmtpcs-daemon
!
```

The changed channel blocks should look like this:

```
!
! tcp_lmtpcs (LMTP client - store)
tcp_lmtpcs defragment lmtp  multigate connectcanonical \
  fileinto @$40:$U+$S@$D port 225 nodns single_sys \
  subdirs 20 maxjobs 7 pool SMTP_POOL dequeue_removeroute
lmtpcs-daemon
```

16.5.1 To Configure Back End Stores with LMTP and a Minimal MTA

The back end stores require only a minimal MTA if they are receiving messages over LMTP. They require a dispatcher, a job controller and a simple MTA configuration. In particular they need a `dispatcher.cnf`, `job_controller.cnf` and a `mappings` file which comprise the only significant part of the MTA configuration.

The `dispatcher.cnf` file must contain the following:

```
! VERSION=1.1
! IMTA default dispatcher configuration file
!
! Global defaults
!
MIN_PROCS=1
MAX_PROCS=10
MIN_CONNS=30
MAX_CONNS=50
MAX_SHUTDOWN=2
MAX_LIFE_TIME=86400
MAX_LIFE_CONNS=10000
MAX_IDLE_TIME=600
HISTORICAL_TIME=0
!
! rfc 2033 LMTP server - store
!
[SERVICE=LMTPSS]
PORT=225
IMAGE=IMTA_BIN:tcp_lmtp_server
```

```
LOGFILE=IMTA_LOG:tcp_lmtppss_server.log
PARAMETER=CHANNEL=tcp_lmtppss
STACKSIZE=2048000
! Uncomment the following line and set INTERFACE_ADDRESS to an
! appropriate host IP (dotted quad) if the dispatcher needs to
! listen on a specific interface (e.g. in a HA environment).
! INTERFACE_ADDRESS=!
! rfc 2033 LMTP server - native
!
```

Note that by default, the LMTP services in the `dispatcher.cnf` file are commented out. You must uncomment them to get LMTP to work.

The normal dispatcher options of `MAX_CONNS`, `MAX_PROCS`, `MAX_LIFE_CONNS`, and `MAX_LIFE_TIME` can also be set, but need to be set appropriately for your hardware.

The `PORT_ACCESS` mapping is important. The LMTP implementation for the back end servers is intended as a private protocol between Sun Java System Messaging Server relays and back end stores. You must use the `PORT_ACCESS` mapping to make sure that only such relays can connect to these services. Your mapping file should look like this:

PORT_ACCESS

```
TCP|*|225|192.18.74.206|* $Y
TCP|*|226|192.18.74.206|* $Y
TCP|*|225|192.18.74.129|* $Y
TCP|*|226|192.18.74.129|* $Y
TCP|*|*|*|* $N500$ Do$ not$ connect$ to$ this$ machine
```

The IP address above are LMTP server and client IP address. You should replace the sample IP addresses specified in the `PORT_ACCESS` mapping table here with the IP addresses of your relays on the network that connect to the back end stores.

There has to be an `imta.cnf` file, but it is there merely to make the configuration complete. A minimal `imta.cnf` file consists of the following channel definitions:

```
!
! IMTA configuration file
!
! tcp_lmtppss (LMTP server - store)
tcp_lmtppss lmtp flagtransfer
tcp_lmtppss-daemon
```

Note that by default, the LMTP channel definitions are commented out. You must uncomment them if you want LMTP to work.

You can use the default `job_controller.cnf` file created on installation. No modification of this file is required.

16.5.2 Configuring Relays for Sending Messages Via LMTP to Back End Systems with Message Stores and Full MTAs

There are situations where you might want the back end stores to have the full capabilities of the MTA but still to have the load savings of using LMTP. For example, you might want program delivery on the back end store. In this case the relays should be configured as described above in “[To Configure the Inbound MTA Relays with LMTP](#)” on page 526

16.5.3 Configuring LMTP on Back End Message Store Systems Having Full MTAs

The only changes from the configuration of a back end store messaging system to one with LMTP direct delivery to the store are that the following lines need to be added to the end of the `dispatcher.cnf` file:

```
! rfc 2033 LMTP server - store
[SERVICE=LMTPSS]
PORT=225
IMAGE=IMTA_BIN:tcp_lmtp_server
LOGFILE=IMTA_LOG:tcp_lmtpss_server.log
PARAMETER=CHANNEL=tcp_lmtpss
STACKSIZE=2048000
! Uncomment the following line and set INTERFACE_ADDRESS to an
! appropriate host IP (dotted quad) if the dispatcher needs to
! listen on a specific interface (e.g. in a HA environment).
!INTERFACE_ADDRESS=
```

Note that by default, the LMTP services in the `dispatcher.cnf` file are commented out. You must uncomment them to get LMTP to work. Also, the LMTP port numbers are just examples, and can be anything you choose.

This is the same as the whole `dispatcher.cnf` file described above for when the back end store is configured only for LMTP. The mappings file also requires the `PORT_ACCESS` mappings as described for LMTP only back end stores.

16.5.4 Handling 4.2.1 Mailbox Busy Error in Response to LMTP Message Data

If the LMTP channel option MAILBOX_BUSY_FAST_RETRY is set to 1 (the default) a 4.2.1 Mailbox busy error in response to LMTP message data is handled by retrying the message after a random but short interval; normal message backoff values do not apply. Setting the option to 0 disables this behavior.

16.6 LMTP Protocol as Implemented

This section provides a sample LMTP dialogue with an explanation of what is seen in that dialogue. The LMTP client on the relay uses standard LMTP protocol to talk to the LMTP server on the back end store. However the protocol is used in specific ways. For example:

```
--> LHLO  
<--- 250 OK
```

No action is taken on the LHLO message. The reply is always 250 OK.

```
--> MAIL FROM: address size=messageSizeInBytes  
<--- 250 OK
```

No checks or conversions are made on the originator address. The `size=` parameter gives a size in bytes for the message that is to be delivered. This is the size of the message exactly as it appears in the protocol. It is not necessarily the exact size of the message, but the actual message size will not exceed this size. The LMTP server allocates a memory buffer of this size to receive the message.

```
--> RCPT TO: uid+folder@domain xquota=size,number xdflg=xxx  
<--- 250 OK
```

No checks are made on the recipient addresses at the time they are received, but a list of recipients is built for later use. Note that the `@domain` part of the address is omitted for `uids` in the primary domain, and that the `+folder` part is optional. This is the same address format used by the message store channel in the MTA.

The `xquota=` parameter gives the user's message quotas which consist of the maximum total size and the maximum number of messages. The MTA provides this information which it retrieves while performing an LDAP lookup on the user to do the address translation. This information is used to keep the quota information in the message store synchronized with the directory. Getting the quota information does not result in an additional performance hit.

The `xdflg=` parameter specifies a number which is interpreted as a bit field. These bits control how the message is delivered. For example, the bit whose value is 2, if set, guarantees delivery of

the message even if the user is over quota. (Note that `xdflg` is an internal parameter and the bits in it are subject to change or addition without notice. We do not support other clients using this extension with our server, nor do we support using our client with some other server and this parameter.)

This interaction may be repeated many times, once for each recipient.

```
-->DATA
--> <the message text>
-->.
```

The LMTP client then sends the entire message, dot-stuffed, just as SMTP does. The message finishes with a dot (.) alone on a line. If the message size is exceeded the LMTP server sends:

```
<--- 500 message too big
```

and ends the connection.

Assuming that the message is received correctly, the LMTP server then sends back to the LMTP client the status for each recipient given in the `RCPT TO:` lines. For instance, if the message is delivered successfully, the response is:

```
<--- 250 2.5.0 address OK
```

where `address` is exactly as it appeared on the `RCPT TO:` line.

The conversation can either repeat with another `MAIL FROM:` line or end with the following interaction:

```
--> quit
<--- 221 OK
```

[Table 16–1](#) shows the possible status codes for each recipient. This three-column table shows the short code in the first column, its long-code equivalent in the second column and the status text in the third column. 2.x.x status codes are success codes, 4.x.x codes are retryable errors, 5.x.x codes are non retryable errors.

TABLE 16–1 LMTP Status Codes for Recipients

Short Code	Long Code	Status Text
250	2.5.0	OK
420	4.2.0	Mailbox Locked
422	4.2.2	Quota Exceeded
420	4.2.0	Mailbox Bad Formats

TABLE 16-1 LMTP Status Codes for Recipients *(Continued)*

Short Code	Long Code	Status Text
420	4.2.0	Mailbox not supported
430	4.3.0	IMAP IOERROR
522	5.2.2	Persistent Quota Exceeded
523	5.2.3	Message too large
511	5.1.1	mailbox nonexistent
560	5.6.0	message contains null
560	5.6.0	message contains nl
560	5.6.0	message has bad header
560	5.6.0	message has no blank line

Otherwise, there are changes to the delivery options for mailbox, native (and, therefore, UNIX), and file. The object of these rules is to generate addresses that will cause the messages to be sent through the appropriate LMTP channel to the back end servers. The addresses generated are source routed addresses of the form:

@sourceroute:localpart@domain

Vacation Automatic Message Reply

For automatically generated responses to email (autoreply), specifically vacation messages, the MTA uses Message Disposition Notifications (*MDNs*) and the Sieve scripting language. MDNs are email messages sent by the MTA to a sender and/or postmaster reporting on a message's delivery disposition. MDNs are also known as read receipts, acknowledgements, receipt notifications, or delivery receipts. The Sieve is a simple scripting language used to create mail filters. Unlike the Messaging Server 5.x, the character set used is UTF-8 instead of ISO-2022-JP.

This section describes the vacation autoreply mechanism. In most cases, modification to the default configuration is unnecessary, but there is a case where you may wish to configure your system such that vacation processing is done on MTA relay machines rather than on back-end message stores.

This chapter consists of the following sections:

- “[17.1 Vacation Autoreply Overview](#)” on page 533
- “[17.2 Configuring Autoreply](#)” on page 534
- “[17.3 Vacation Autoreply Theory of Operation](#)” on page 536
- “[17.4 Vacation Autoreply Attributes](#)” on page 537
- “[17.5 Other Auto Reply Tasks and Issues](#)” on page 539

17.1 Vacation Autoreply Overview

Vacation Sieve scripts are generated automatically from the various LDAP vacation attributes (see “[17.4 Vacation Autoreply Attributes](#)” on page 537). They can also be specified explicitly for additional flexibility. The underlying mechanism for tracking vacations is a set of files, one per intended recipient, that keep track of when replies were sent to the various senders.

Note – charset of vacation message has changed to UTF-8.

By default, the MTA evaluates vacation on the back-end store systems. However, since MTA relays do not do as much work as back-end stores, for performance reasons, you can have the MTA evaluate vacation on the mail relay machines instead of on the back-end store. Use of this feature, however, can result in vacation responses being sent out more often than intended because different relays handle different messages. If you do not want vacation messages to be sent out more often than you intend, you may share the tracking of files between the relays. If this is also unacceptable to you, you can always have vacation evaluated on the back-end store systems.

17.2 Configuring Autoreply

Delivery addresses are generated through a set of patterns. The patterns used depend upon the values defined for the `mailDeliveryOption` attribute. A delivery address is generated for each valid `mailDeliveryOption`. The patterns are defined by the MTA option `DELIVERY_OPTIONS`, which are defined in the `option.dat` file. The default autoreply rule in `DELIVERY_OPTIONS` in the `option.dat` file is:

```
*^!autoreply=$M+$D@bitbucket
```

The MTA notes the “`^`” on the autoreply `DELIVERY_OPTION` MTA option. This causes the MTA to check the vacation dates. If the current date is within the vacation dates, processing continues and the MTA notes the “`!`” on the autoreply `DELIVERY_OPTION`. The MTA then creates a vacation Sieve script based on the various autoreply LDAP attributes on the user’s entry. The autoreply rule can have the prefix characters “`!`”, “`#`”, “`^`”, and “`*`”.

You could have the “`!`” flag on the mailbox delivery option. This would enable the generation of the vacation script unconditionally. It makes sense, however, to have the autoreply machinery enabled by a separate delivery option so that it can be further gated by the “`^`” flag. Checking the dates at this stage is more efficient than engaging the Sieve logic.

Table 17–1 shows the prefix characters used for the autoreply rule in the first column and their definitions in the second column.

TABLE 17–1 Prefix Characters for the Autoreply Rule in `DELIVERY_OPTIONS`

Prefix Character	Definition
!	Enable the generation of the autoreply Sieve script.
#	Allows the processing to take place on relays.
^	Option is only evaluated if the vacation dates indicate that it should be evaluated.

TABLE 17-1 Prefix Characters for the Autoreply Rule in DELIVERY_OPTIONS *(Continued)*

Prefix Character	Definition
@	Extract preferred language information from various message header fields as well as from LDAP entries associated with envelope From: addresses. For this information to be available at the correct time, the message must pass through the reprocess channel when autoreply is engaged. This is done by adding the @ flag to the autoreply delivery option. Note that the addition of a channel hop increases message processing overhead.

The autoreply rule itself specifies an address destined for the bitbucket channel. The mail is considered delivered by this method once the autoreply is generated, but the MTA machinery requires a delivery address. Anything delivered to the bitbucket channel is discarded.

17.2.1 Configuring Autoreply on the Back-end Store System

The default autoreply rule in DELIVERY_OPTIONS causes the autoreply to take place on the mail server that serves the user. If you want vacation messages to be evaluated on the back-end store system, you do not have to configure anything. This is the default behavior.

▼ To Configure Autoreply on a Relay

If you want to evaluate vacation on the relay rather than on the back-end store system to enhance performance, edit the option.dat file and prepend the character # to the autoreply rule in DELIVERY_OPTIONS.

- 1 Use an editor to open the option.dat file.
- 2 Add or change the DELIVERY_OPTIONS option so the autoreply rule now looks like:

```
#*^!autoreply=$M+$D@bitbucket
```

The default DELIVERY_OPTIONS option looks like:

```
DELIVERY_OPTIONS=*mailbox=$M%$\\$2I$_+$2S@ims-ms-daemon, \
&members=*, \
*native=$M@native-daemon, \
/hold=@hold-daemon:$A, \
*unix=$M@native-daemon, \
&file=+$F@native-daemon, \
&@members_offline=* \
,program=$M%$P@pipe-daemon, \
#forward=**, \
*^!autoreply=$M+$D@bitbucket
```

This allows the processing to take place on relays. If you have the MTA perform autoreplies on the relays, then either each relay can keep track independently of whether a particular correspondent has sent an away message recently, or this information can be shared between

the relays. The former case is simpler, especially if having away messages sent out too many times does not matter. If you want strict application of the away message frequency rules, then the information must be shared between relays. To share the information among the relays, the files should be NFS mounted. For important information on NFS-mounting, see “[12.8.2.3 Using NFS-based File Systems for Defragmentation and Vacation Caching](#)” on page 408

The location of these files is controlled by the option VACATION_TEMPLATE. This option (in option.dat) should be set to <path>%A where <path> is the path to a directory that is shared between the various relay machines. The template needs to be a file:URL and you use \$U to substitute the name of the user. The default setting is:

```
VACATION_TEMPLATE=file:///opt/SUNWmsgsr/data/vacation/$3I/$1U/$2U/$U.vac
```

See [Table 9–6](#) for metacharacter descriptions.

Note – Vacation file templates now have access to the UID, allowing paths to vacation files to be built based on the user's UID. Additionally, the address used in determining the vacation file path is now the one stored in the user's mail attribute; previously the current recipient address was used.

17.3 Vacation Autoreply Theory of Operation

The vacation action, when invoked, works as follows:

1. The Sun Java System Messaging Server checks to make sure that the vacation action was performed by a user level rather than a system level Sieve script. An error results if vacation is used in a system level script.
2. The “no vacation notice” internal MTA flag is checked. If it is set, processing terminates and no vacation notice is sent.
3. The return address for the message is now checked. If it is blank, processing terminates and no vacation notice is sent.
4. The MTA checks to see if the user's address or any of the additional addresses specified in the :addresses tagged argument appear in a To:, Cc:, Resent-to:, or Resent-cc: header field for the current message. Processing terminates and no vacation notice is sent if none of the addresses is found in any of the header fields.
5. The Messaging Server constructs a hash of the :subject argument and the reason string. This string, along with the return address of the current message, is checked against a per-user record of previous vacation responses. Processing terminates and no response is sent if a response has already been sent within the time allowed by the :days argument.
6. The Messaging Server constructs a vacation notice from the :subject argument, reason string, and :mime argument. Two basic forms for this response message are possible:
 - A message disposition notification of the form specified in RFC 2298, with the first part containing the reason text.

- A single part text reply. (This form is only used to support the “reply” autoreply mode attribute setting.)

Note that `mailautoreplymode` is automatically set to `reply` when vacation messages are configured through Messenger Express.

The “no vacation notice” MTA flag is clear by default. It can be set by a system level Sieve script through the use of the nonstandard `novacation` action. The `novacation` Sieve action is only allowed in a system level Sieve script. It will generate an error if it is used in a user level script. You can use this action to implement site-wide restrictions on vacation replies such as blocking replies to addresses containing the substring “MAILER-DAEMON”.

Per-user per-response information is stored in a set of flat text files, one per local user. The location and naming scheme for these files is specified by the setting of the `VACATION_TEMPLATE_MTA` option. This option should be set to a `file: URL`.

Maintenance of these files is automatic and controlled by the `VACATION_CLEANUP` integer MTA option setting. Each time one of these files is opened, the value of the current time in seconds modulo this value is computed. If the result is zero the file is scanned and all expired entries are removed. The default value for the option is 200, which means that there is 1-in-200 chance that a cleanup pass will be performed.

The machinery used to read and write these flat text files is designed in such a way that it should be able to operate correctly over NFS. This allows multiple MTAs to share a single set of files on a common file system.

17.4 Vacation Autoreply Attributes

The set of user LDAP directory attributes that the vacation action uses are:

- Attribute defined by the MTA option `LDAP_AUTOREPLY_ADDRESSES`
This attribute provides the ability to generate `:addresses` arguments to sieve `vacation`. This option has no value by default. The attribute can be multivalued, with each value specifying a separate address to pass to the `:addresses` vacation parameter.
- Attribute defined by `LDAP_PERSONAL_NAME`
Alias processing keeps track of personal name information specified in this attribute and will use this information to construct `From:` fields for any MDNs or vacation replies that are generated. Use with caution to avoid exposing personal information.
- `vacationStartDate`
Vacation start date and time. The value is in the format `YYYYMMDDHHMMSSZ`. This value is normalized to GMT. An autoreply should only be generated if the current time is after the time specified by this attribute. No start date is enforced if this attribute is missing. The MTA can be instructed to look at a different attribute for this information by setting the `LDAP_START_DATE` MTA option to a different attribute name.

This attribute will be read and checked by the code that generated the Sieve script. Vacation processing will be aborted if the current date is before the vacation start date. This attribute cannot be handled by the script itself because at present Sieve lacks date/time testing and comparison facilities.

- **vacationEndDate**

Vacation end date and time. The value is in the format YYYYMMDDHHMMSSZ. This value is normalized to GMT. An autoreply should only be generated if the current time is before the time specified by this attribute. No end date is enforced if this attribute is missing. The MTA can be instructed to look at a different attribute for this information by setting the `LDAP-END-DATE` MTA option to a different attribute name.

This attribute will be read and checked by the code that generated the Sieve script. Vacation processing will be aborted if the current date is after the vacation end date. This attribute cannot be handled in the script itself because at present Sieve lacks date/time testing and comparison facilities.

- **mailAutoReplyMode**

Specifies autoreply mode for the user mail account. Valid values of this attribute are:

- `echo` - Create a multipart that echoes the original message text in addition to the added `mailAutoReplyText` or `mailAutoReplyTextInternal` text.
- `reply` - Send a single part reply as specified by either `mailAutoReplyText` or `mailAutoReplyTextInternal` to the original sender.

These modes will appear in the Sieve script as nonstandard :`echo` and :`reply` arguments to the vacation action. `echo` will produce a “processed” message disposition notification (MDN) that contains the original message as returned content. `reply` will produce a pure reply containing only the reply text. An illegal value will not manifest as any argument to the vacation action and this will produce an MDN containing only the headers of the original message. Note also that selecting an autoreply mode of `echo` causes an automatic reply to be sent to every message regardless of how recently a previous reply was sent.

The MTA can be instructed to use a different attribute for this information by setting the `LDAP-AUTOREPLY-MODE` MTA option to a different attribute name.

- **mailAutoReplySubject**

Specifies the contents of the subject field to use in the autoreply response. This must be a UTF-8 string. This value gets passed as the :`subject` argument to the vacation action. The MTA can be instructed to use a different attribute for this information by setting the `LDAP-AUTOREPLY-SUBJECT` MTA option to a different attribute name.

- **mailAutoReplyText**

Autoreply text sent to all senders except users in the recipient's domain. If not specified, external users receive no vacation message. The MTA can be instructed to use a different attribute for this information by setting the `LDAP-AUTOREPLY-TEXT` MTA option to a different attribute name.

- `mailAutoReplyTextInternal`

Auto-reply text sent to senders from the recipients domain. If not specified, then internal users get the mail autoreply text message. The MTA can be instructed to use a different attribute for this information by setting the `LDAP_AUTOREPLY_TEXT_INT` MTA option to a different attribute name.

The MTA will pass either the `mailAutoReplyText` or `mailAutoReplyTextInternal` attribute value as the reason string to the vacation action.

- `mailAutoReplyTimeOut`

Duration, in hours, for successive autoreply responses to any given mail sender. Used only when `mailAutoReplyMode=reply`. If value is 0 then a response is sent back every time a message is received. This value will be converted to the nonstandard :`hours` argument to the vacation action. (Normally the Sieve vacation action only supports the :`days` argument for this purpose and does not allow a value of 0.)

If this attribute doesn't appear on a user entry, a default time-out will be obtained from the `AUTOREPLY_TIMEOUT_DEFAULT` MTA option. The MTA can be instructed to use a different attribute for this information by setting the `LDAP_AUTOREPLY_TIMEOUT` MTA option.

The MTA can choose between multiple LDAP attributes and attribute values with different language tags and determine the correct value to use. The language tags in effect are compared against the preferred language information associated with the envelope from address. Currently the only attributes receiving this treatment are `LDAP_AUTOREPLY SUBJECT` (normally `mailAutoReplySubject`), `LDAP_AUTOREPLY_TEXT` (normally `mailAutoReplyText`), `LDAP_AUTOREPLY_TEXT_INT` (normally `mailAutoReplyTextInternal`), `LDAP_SPARE_4`, `LDAP_SPARE_5`, `LDAP_PREFIX_TEXT` and `LDAP_SUFFIX_TEXT`.

It is expected that each attribute value will have a different language tag value. If different values have the same tag value the choice between them will be essentially random.

17.5 Other Auto Reply Tasks and Issues

This section describes auto reply tasks and issues not described in the configuration section.

17.5.1 To Send Autoreply Messages for Email That Have Been Automatically Forwarded from Another Mail Server

An autoreply problem can occur when the MTA receives a message that has been automatically forwarded from another system in some other administrative domain. For example, if a customer has a home account with `sesta.com` and the customer sets that account to automatically forward messages to their work account at `siroe.com` and if `siroe.com` uses Messaging Server and that user has set his account to autoreply a vacation message, then Messaging Server will have a problem sending out a vacation message.

The problem occurs because the `sesta.com` mail server changes the envelope `To:` address from `user@sesta.com` to `user@siroe.com`, but it will not change the `To:` header, which remains `user@sesta.com`. When the MTA receives the message, it looks at the header address only. It attempts to match this address with an address in the LDAP user directory. If it finds a match with someone who has set autoreply, then a vacation message is sent. Because there is no LDAP address match to `user@sesta.com`, no vacation message is sent. The problem is that the actual address is in the envelope and not in the header.

Since the recipient's address known to the remote system doing automatic forwarding isn't known to correspond to the user by the local system, there needs to be a way for the recipient to make such addresses known to the local system so vacation replies will be sent when necessary.

The `:addresses` argument to the Sieve `vacation` action provides this capability. It accepts a list of addresses that correspond to the recipient for purposes of making this check. The attribute defined by the MTA option `LDAP_AUTOREPLY_ADDRESSES` allows specification of such addresses in the user's LDAP entry.

To provide autoreply capability for messages that have been automatically forwarded from mail servers in some other administrative domain, the user or administrator would set the email addresses from where those messages may be forwarded to the attribute defined by `LDAP_AUTOREPLY_ADDRESSES`.

Mail Filtering and Access Control

This chapter discusses how to filter mail based on its source (sender, IP address and so on) or header strings. Two mail filtering mechanisms are used, controlling access to the MTA using mapping tables and Sieve server-side rules (SSR).

Limiting access to the MTA using the mapping tables allows messages to be filtered based on From: and To: addresses, IP address, port numbers and source or destination channels.

Mapping tables allow SMTP relaying to be enabled or disabled. Sieve is a mail filtering script that allows messages to be filtered based on strings found in headers. (It doesn't work on the message body.)

If envelope-level controls are desired, use mapping tables to filter mail. If header-based controls are desired, use Sieve server-side rules.

This chapter is divided into two parts:

[“18.1 PART 1. MAPPING TABLES” on page 541](#). Allows the administrator to control access to MTA services by configuring certain mapping tables. The administrator can control who can or cannot send mail, receive mail through Messaging Server.

[“18.9 PART 2. MAILBOX FILTERS” on page 568](#). Allows users and administrators to filter messages and specify actions on those filtered messages based on a string found in the message header. Uses the Sieve filter language and can filter at the channel, MTA or user level.

18.1 PART 1. MAPPING TABLES

Part 1 contains the following sections:

- [“18.2 Controlling Access with Mapping Tables” on page 542](#)
- [“18.3 Access Control Mapping Table Flags” on page 544](#)
- [“18.4 When Access Controls Are Applied” on page 556](#)
- [“18.5 To Test Access Control Mappings” on page 556](#)

- “18.6 To Add SMTP Relaying” on page 557
- “18.7 Configuring SMTP Relay Blocking” on page 560
- “18.8 Handling Large Numbers of Access Entries” on page 566

18.2 Controlling Access with Mapping Tables

You can control access to your mail services by configuring certain mapping tables. These mapping tables allow you to control who can or cannot send mail, receive mail, or both.

Table 18–1 lists the mapping tables described in this section. The application information string supplied to the FROM_ACCESS, MAIL_ACCESS, and ORIG_MAIL_ACCESS mappings includes the system name claimed in the HELO/EHLO SMTP command. This name appears at the end of the string and is separated from the rest of the string (normally “SMTP*”) by a slash. The claimed system name can be useful in blocking some worms and viruses.

18.2.1 Access Control Mapping Tables—Operation

Like all mapping tables, access control mapping tables have the same general format (see “[10.3 Mappings File](#)” on page 237). They consist of a mapping table name, followed by a line break, followed by one or more mapping entries. Mapping entries consist of a *search pattern* on the left side and a *template* on the right side. The search pattern filters specific messages and the template specifies actions to take on the message. For example:

```
SEND_ACCESS

*|Elvis1@sesta.com|*|*      $Y
*|Nelson7@sesta.com|*|*      $Y
*|AkiraK@sesta.com|*|*      $Y
*|*@sesta.com|*|*          $NMail$ Blocked
```

In this example all email from the domain `sesta.com` except those of `Elvis1`, `Nelson`, `AkiraK` are blocked.

The search pattern for access control mapping entries consist of a number of search criteria separated by vertical bars (|). The order of the search criteria depends on the access mapping table and is described in subsequent sections. But as an example, the `SEND_ACCESS` mapping table has the following search form:

src-channel|from-address|dst-channel|to-address

where *src-channel* is the channel queueing the message; *from-address* is the address of the message's originator; *dst-channel* is the channel to which the message will be queued; and *to-address* is the address to which the message is addressed. Use of an asterisk in any of these four fields causes that field to match any channel or address, as appropriate.

Note – Whenever the `mappings` file is modified, you must recompile the configuration (see “[10.1 Compiling the MTA Configuration](#)” on page 233).

TABLE 18-1 Access Control Mapping Tables

Mapping Table	Description
<code>SEND_ACCESS</code> (See “ 18.3.1 SEND_ACCESS and ORIG_SEND_ACCESS Tables ” on page 546.)	Used to block incoming connections based on envelope <code>From</code> address, envelope <code>To</code> address, source and destination channels. The <code>To</code> address is checked after rewriting, alias expansion, and so on, have been performed.
<code>ORIG_SEND_ACCESS</code> (See “ 18.3.1 SEND_ACCESS and ORIG_SEND_ACCESS Tables ” on page 546.)	Used to block incoming connections based on envelope <code>From</code> address, envelope <code>To</code> address, source and destination channels. The <code>To</code> address is checked after rewriting but before alias expansion.
<code>MAIL_ACCESS</code> (See “ 18.3.2 MAIL_ACCESS and ORIG_MAIL_ACCESS Mapping Tables ” on page 548.)	Used to block incoming connections based on combined information found in <code>SEND_ACCESS</code> and <code>PORT_ACCESS</code> tables: that is, the channel and address information found in <code>SEND_ACCESS</code> combined with the IP address and port number information found in <code>PORT_ACCESS</code> .
<code>ORIG_MAIL_ACCESS</code> (See “ 18.3.2 MAIL_ACCESS and ORIG_MAIL_ACCESS Mapping Tables ” on page 548.)	Used to block incoming connections based on combined information found in <code>ORIG_SEND_ACCESS</code> and <code>PORT_ACCESS</code> tables: that is, the channel and address information found in <code>ORIG_SEND_ACCESS</code> combined with the IP address and port number information found in <code>PORT_ACCESS</code> .
<code>FROM_ACCESS</code> (See “ 18.3.3 FROM_ACCESS Mapping Table ” on page 550.)	Used to filter mail based on envelope <code>From</code> addresses. Use this table if the <code>To</code> address is irrelevant.
<code>PORT_ACCESS</code> (See “ 18.3.4 PORT_ACCESS Mapping Table ” on page 552.)	Used to block incoming connections based on IP number.
<code>IP_ACCESS</code>	Used to block incoming connections based on source channel, IP address count for remote server, index of current IP address being tried. See “ 18.3.5 IP_ACCESS Mapping Table ” on page 554.

The `MAIL_ACCESS` and `ORIG_MAIL_ACCESS` mappings are the most general, having available not only the address and channel information available to `SEND_ACCESS` and `ORIG_SEND_ACCESS`, but also any information that would be available via the `PORT_ACCESS` mapping table, including IP address and port number information.

18.3 Access Control Mapping Table Flags

This section consists of the following subsections:

- “18.3.1 SEND_ACCESS and ORIG_SEND_ACCESS Tables” on page 546
- “18.3.2 MAIL_ACCESS and ORIG_MAIL_ACCESS Mapping Tables” on page 548
- “18.3.3 FROM_ACCESS Mapping Table” on page 550
- “18.3.4 PORT_ACCESS Mapping Table” on page 552
- “18.3.5 IP_ACCESS Mapping Table” on page 554
- “18.3.6 To Limit Specified IP Address Connections to the MTA” on page 555

Table 18–2 shows the access mapping flags and metacharacters relevant for the SEND_ACCESS, ORIG_SEND_ACCESS, MAIL_ACCESS, ORIG_MAIL_ACCESS, and FROM_ACCESS mapping tables. Note that the PORT_ACCESS mapping table, supports a somewhat different set of flags (see [Table 18–3](#)).

Flags with arguments must have those arguments arranged in the reading order shown in the table. For example:

ORIG_SEND_ACCESS

```
tcp_local|*|tcp_local|*      $N$D30|Relaying$ not$ allowed
```

In this case, the proper order is the delay period followed by the rejection string. Note that the flags themselves can be in any order. So the following entries have identical results:

```
30|Relaying$ not$ allowed$D$N
$N30|Relaying$ not$ allowed$D
30|$N$DRelaying$ not$ allowed
```

TABLE 18–2 Access Mapping Flags and Metacharacters

Flag	Description
\$A	Set if SASL has been used. See “ Check for Special Flags ” on page 247
\$B	Redirect the message to the bitbucket.
\$D	Set if delay delivery receipts requests (not available in FROM_ACCESS). See “ Check for Special Flags ” on page 247.
\$E	Set if An EHLO command was issued/accepted and therefore ESMTP was used (not available in FROM_ACCESS). See “ Check for Special Flags ” on page 247.
\$F	Set if failure delivery receipts requested (not available in FROM_ACCESS). See “ Check for Special Flags ” on page 247.
\$H	Hold the message as a .HELD file.

TABLE 18-2 Access Mapping Flags and Metacharacters *(Continued)*

Flag	Description
\$L	Set if LMTP was used (not available in FROM_ACCESS). See “Check for Special Flags” on page 247.
\$S	Set if success delivery receipts requested (not available in FROM_ACCESS). See “Check for Special Flags” on page 247.
\$T	Set if TLS has been used. See “Check for Special Flags” on page 247
\$U	If used in ORIG_SEND_ACCESS, SEND_ACCESS, ORIG_MAIL_ACCESS, and MAIL_ACCESS, takes a single integer argument from the beginning of the mapping and sets the value of MM_DEBUG accordingly. Additionally, channel-level debugging is also enabled if possible. The result is that you enable debugging based on items such as source IP address, original address, recipient address, and so on.
\$Y	Allow access.
\$V	Causes a forced discard to be performed for all recipient(s).
\$Z	Causes a forced jettison to be performed for all recipient(s).
\$!	Available in FROM_ACCESS only. Disables the sending of vacation messages regarding this message; that is, it sets the novacation flag. (This is the same effect as setting novacation explicitly in a system/channel Sieve.) This will override (prevent application) of a subsequent vacation action that would otherwise have applied to the message.

Flags with Arguments, in Argument Reading Order+ (DO NOT ALPHABETIZE THIS LIST!)

\$UInteger	Takes a single integer argument from the beginning of the mapping and sets MM_DEBUG accordingly. Additionally, channel-level debugging is also enabled if possible. The result is that debugging can now be enabled based on things like source IP address, original address, recipient address, and so on.
\$Jaddress	* Replace original envelope From: address with specified <i>address</i> .
\$Kaddress	* ++ Replace original Sender: address with specified <i>address</i> .
\$Iuser identifier	Check specified user for group ID.
\$<string>	+++ Send <i>string</i> to syslog (UNIX, user.notice facility and severity) or to the event log (NT) if probe matches.
\$>string	+++ Send <i>string</i> to syslog (UNIX, user.notice facility and severity) or to the event log (NT) if access is rejected.
\$Ddelay	Delay response for an interval of <i>delay</i> hundredths of seconds; a positive value causes the delay to be imposed on each command in the transaction; a negative value causes the delay to be imposed only on the address handover (SMTP MAIL FROM: command for the FROM_ACCESS table; SMTP RCPT TO: command for the other tables).
\$Ttag	Prefix with <i>tag</i> .
\$Aheader	Add the header line <i>header</i> to the message.

TABLE 18–2 Access Mapping Flags and Metacharacters *(Continued)*

Flag	Description
\$conversion_tag	If used in ORIG_SEND_ACCESS, SEND_ACCESS, ORIG_MAIL_ACCESS, and MAIL_ACCESS, it reads a value from the mapping result and treats it as a set of conversion tags to be applied to the current recipient. If used with FROM_ACCESS, conversion tags are applied to all recipients. \$G is positioned after \$A (header address) in the sequence of arguments read from the mappings. See “ Mail Conversion Tags ” on page 440
\$Sx,y,z	* Causes an additional separated argument to be read from the mapping result. This argument consists of one to three integer values separated by commas. The first value establishes a new minimum blocklimit for the transaction, the second establishes a new minimum recipientlimit, and the third a new minimum recipientcutoff. The argument is read from the mapping result after any capture argument has been read. See “ 12.9.2 Specifying Absolute Message Size Limits ” on page 411
\$Xerror-code	Lets you specify the extended SMTP <i>error-code</i> (the digit.digit.digit part), and if the first digit is a 4 rather than a 5, then you’ll get a 452 SMTP temporary error, rather than the usual 550 SMTP permanent error. For example:
	ORIG_SEND_ACCESS <...probe...> \$N\$X4.5.9 Temporary\$ problem\$ with\$ address;\$ try\$ later\$
\$,spamadjust_arg	Allows you to perform a sieve spamadjust operation from the access mapping tables. Argument takes the same form as a spamadjust argument. Note also that some of these mappings are applied on a per-recipient basis. Any spamadjust operation that is done applies to all recipients.
\$Nstring	Reject access with the optional error text <i>string</i> .
\$Fstring	Synonym for \$N string; that is, reject access with the optional error text <i>string</i> .

* Available for FROM_ACCESS table only.

+ To use multiple flags with arguments, separate the arguments with the vertical bar character, |, placing the arguments in the order listed in this table.

++ For the \$K flag to take effect in the FROM_ACCESS mapping table, the source channel must include the authrewrite keyword.

+++ It is a good idea to use the \$D flag when dealing with problem senders, to prevent a denial of service attack. In particular, it is a good idea to use \$D in any \$> entry or \$< entry rejecting access.

18.3.1 SEND_ACCESS and ORIG_SEND_ACCESS Tables

You can use the SEND_ACCESS and ORIG_SEND_ACCESS mapping tables to control who can or cannot send mail, receive mail, or both. The access checks have available a message’s envelope From: address and envelope To: addresses, and knowledge of what channel the message came in, and what channel it would attempt to go out.

If a SEND_ACCESS or ORIG_SEND_ACCESS mapping table exists, then for each recipient of every message passing through the MTA, the MTA will scan the table with a string of the following form (note the use of the vertical bar character, |):

src-channel|from-address|dst-channel|to-address

The *src-channel* is the channel queueing the message; *from-address* is the address of the message's originator; *dst-channel* is the channel to which the message will be queued; and *to-address* is the address to which the message is addressed. Use of an asterisk in any of these four fields causes that field to match any channel or address, as appropriate.

The addresses here are envelope addresses; that is, envelope From: address and envelope To: address. In the case of SEND_ACCESS, the envelope To: address is checked after rewriting, alias expansion, etc., have been performed; in the case of ORIG_SEND_ACCESS the originally specified envelope To: address is checked after rewriting, but before alias expansion.

If the search string matches a pattern (that is, the left-hand side of an entry in the table), then the resulting output of the mapping is checked. If the output contains the flags \$Y or \$y, then the enqueue for that particular To: address is permitted. If the output contains any of the flags \$N, \$n, \$F, or \$f, then the enqueue to that particular address is rejected. In the case of a rejection, optional rejection text may be supplied in the mapping output. This string will be included in the rejection error the MTA issues. If no string is output (other than the \$N, \$n, \$F or \$f flag), then default rejection text will be used. For descriptions of additional flags, see “[18.3 Access Control Mapping Table Flags](#)” on page 544.

Setting the MTA option ACCESS_ORCPT to 1 adds an additional vertical bar delimited field to the probe value passed to the SEND_ACCESS, ORIG_SEND_ACCESS, MAIL_ACCESS, and ORIG_MAIL_ACCESS mapping tables that contains the original recipient (ORCPT) address. If the message doesn't have an ORCPT address, the original unmodified RCPT TO: address is used instead. The default is 0, and the probe value is at the end:

src-channel|from-address|dst-channel|to-address|ORCPT_address

In the following example, mail sent from UNIX user agents such as mail, Pine, and so on, originates from the local, l, channel and messages to the Internet go out a TCP/IP channel of some sort. Suppose that local users, with the exception of the postmaster, are not allowed to send mail to the Internet but can receive mail from there. Then the SEND_ACCESS mapping table shown in the example below is one possible way to enforce this restriction. In the mapping table, the local host name is assumed to be `sesta.com`. In the channel name “`tcp_*`”, a wild card is used so as to match any possible TCP/IP channel name (for example, `tcp_local`).

EXAMPLE 18-1 SEND_ACCESS Mapping Table

SEND_ACCESS

```
*|postmaster@sesta.com|*|*      $Y
*|*|*|postmaster@sesta.com      $Y
l|*@sesta.com|tcp_*|*          $NInternet$ postings$ are$ not$ permitted
```

In the rejection message, dollar signs are used to quote spaces in the message. Without those dollar signs, the rejection would be ended prematurely and only read “Internet” instead of “Internet postings are not permitted.” Note that this example ignores other possible sources of “local” postings such as from PC-based mail systems or from POP or IMAP clients.

Note – The client attempting to send the message determines whether the MTA rejection error text is actually presented to the user who attempted to send the message. If SEND_ACCESS is used to reject an incoming SMTP message, the MTA merely issues an SMTP rejection code including the optional rejection text; it is up to the sending SMTP client to use that information to construct a bounce message to send back to the original sender.

18.3.2 MAIL_ACCESS and ORIG_MAIL_ACCESS Mapping Tables

The MAIL_ACCESS mapping table is a superset of the SEND_ACCESS and PORT_ACCESS mapping tables. It combines both the channel and address information of SEND_ACCESS with the IP address and port number information of PORT_ACCESS. Similarly, the ORIG_MAIL_ACCESS mapping table is a superset of the ORIG_SEND_ACCESS and PORT_ACCESS mapping tables. The format for the probe string for MAIL_ACCESS is:

port-access-probe-info|app-info|submit-type|send_access-probe-info

Similarly, the format for the probe string for ORIG_MAIL_ACCESS is:

port-access-probe-info|app-info|submit-type|orig_send_access-probe-info

Here *port-access-probe-info* consists of all the information usually included in a PORT_ACCESS mapping table probe in the case of incoming SMTP messages; otherwise, it is blank. *app-info* includes the system name claimed in the HELO/EHLO SMTP command. This name appears at the end of the string and is separated from the rest of the string (normally “SMTP*”) by a slash. The claimed system name can be useful in blocking some worms and viruses. *submit-type* may be one of MAIL, SEND, SAML, or SOML, corresponding to how the message was submitted into Messaging Server. Normally the value is MAIL, meaning it was submitted as a message; SEND, SAML, or SOML can occur in the case of broadcast requests (or combined broadcast/message requests) submitted to the SMTP server. And for the MAIL_ACCESS mapping, *send-access-probe-info* consists of all the information usually included in a SEND_ACCESS mapping table probe. Similarly for the ORIG_MAIL_ACCESS mapping, *orig-send-access-probe-info* consists of all the information usually included in an ORIG_SEND_ACCESS mapping table probe.

Setting the MTA option ACCESS_ORCPT to 1 adds an additional vertical bar delimited field to the probe value passed to the SEND_ACCESS, ORIG_SEND_ACCESS, MAIL_ACCESS, and ORIG_MAIL_ACCESS mapping tables that contains the original recipient (ORCPT) address. If the

message doesn't have an ORCPT address, the original unmodified RCPT TO: address is used instead. The default is 0, and the probe value is at the end. Example:

```
port-access-probe-info|app-info|submit-type|send_access-probe-info|ORCPT_address
```

Having the incoming TCP/IP connection information available in the same mapping table as the channel and address information makes it more convenient to impose certain sorts of controls, such as enforcing what envelope From: addresses are allowed to appear in messages from particular IP addresses. This can be desirable to limit cases of email forgery, or to encourage users to configure their POP and IMAP clients' From: address appropriately. For example, a site that wishes to allow the envelope From: address `vip@siroe.com` to appear only on messages coming from the IP address 1.2.3.1 and 1.2.3.2, and to ensure that the envelope From: addresses on messages from any systems in the 1.2.0.0 subnet are from `siroe.com`, might use a MAIL_ACCESS mapping table as shown in the example below.

EXAMPLE 18-2 MAIL_ACCESS Mapping Table

```
MAIL_ACCESS

! Entries for vip's two systems
!
TCP|*|25|1.2.3.1|*|SMTP*|MAIL|tcp_*|vip@siroe.com|*|* $Y
TCP|*|25|1.2.3.2|*|SMTP*|MAIL|tcp_*|vip@siroe.com|*|* $Y
!
! Disallow attempts to use vip's From: address from other
! systems
!
TCP|*|25|*|*|SMTP*|MAIL|tcp_*|vip@siroe.com|*|* \
$N500$ Not$ authorized$ to$ use$ this$ From:$ address
!
! Allow sending from within our subnet with siroe.com From:
! addresses
!
TCP|*|25|1.2.*.*|*|SMTP*|MAIL|tcp_*|*@siroe.com|*|* $Y
!
! Allow notifications through
!
TCP|*|25|1.2.*.*|*|SMTP*|MAIL|tcp_*||*|*|* $Y
!
! Block sending from within our subnet with non-siroe.com
! addresses
!
TCP|*|25|1.2.*.*|*|SMTP*|MAIL|tcp_*|*|*|* \
$NOnly$ siroe.com$ From:$ addresses$ authorized
```

18.3.3

FROM_ACCESS Mapping Table

The FROM_ACCESS mapping table may be used to control who can send mail, or to override purported From: addresses with authenticated addresses, or both.

The input probe string to the FROM_ACCESS mapping table is similar to that for a MAIL_ACCESS mapping table, minus the destination channel and address, and with the addition of authenticated sender information, if available. Thus, if a FROM_ACCESS mapping table exists, then for each attempted message submission, Messaging Server will search the table with a string of the form (note the use of the vertical bar character, |):

port-access-probe-info|app-info|submit-type|src-channel|from-address|auth-from

Here *port-access-probe-info* consists of all the information usually included in a PORT_ACCESS mapping table probe in the case of incoming SMTP messages; otherwise, it is blank. *app-info* includes the system name claimed in the HELO/EHLO SMTP command. This name appears at the end of the string and is separated from the rest of the string (normally “SMTP*”) by a slash. The claimed system name can be useful in blocking some worms and viruses. *submit-type* may be one of MAIL, SEND, SAML, or SOML, corresponding to how the message was submitted into the MTA. Normally the value is MAIL, meaning it was submitted as a message; SEND, SAML, or SOML can occur in the case of broadcast requests (or combined broadcast/message requests) submitted to the SMTP server. *src-channel* is the channel originating the message (that is, queueing the message); *from-address* is the address of the message's purported originator; and *auth-from* is the authenticated originator address, if such information is available, or blank if no authenticated information is available.

If the probe string matches a pattern (that is, the left-hand side of an entry in the table), the resulting output of the mapping is checked. If the output contains the flags \$Y or \$y, then the enqueue for that particular To: address is permitted. If the output contains any of the flags \$N, \$n, \$F, or \$f, then the enqueue to that particular address is rejected. In the case of a rejection, optional rejection text may be supplied in the mapping output. This string will be included in the rejection error Messaging Server issues. If no string is output (other than the \$N, \$n, \$F, or \$f flag), then default rejection text will be used. For descriptions of additional flags, see “[18.3 Access Control Mapping Table Flags](#)” on page 544.

Besides determining whether to allow a message to be submitted based on the originator, FROM_ACCESS can also be used to alter the envelope From: address via the \$J flag, or to modify the effect of the authrewrite channel keyword (adding a Sender: header address on an accepted message) via the \$K flag. For instance, this mapping table can be used to cause the original envelope From: address to simply be replaced by the authenticated address.

EXAMPLE 18-3 FROM_ACCESS Mapping Table

FROM_ACCESS

|SMTP|*|tcp_auth|*| \$Y

EXAMPLE 18-3 FROM_ACCESS Mapping Table (*Continued*)

```
*|SMTP*|*|tcp_auth|*|*      $Y$J$4
```

When using the **FROM_ACCESS** mapping table to modify the effect on having **authrewrite** set to a nonzero value on some source channel, it is not necessary to use **FROM_ACCESS** if the authenticated address is going to be used verbatim.

For example, with **authrewrite** 2 set on the **tcp_local** channel, the following **FROM_ACCESS** mapping table would not be necessary because **authrewrite** alone is sufficient to get this effect (adding the authenticated address verbatim):

FROM_ACCESS

```
*|SMTP*|*|tcp_auth|*|      $Y
*|SMTP*|*|tcp_auth|*|*    $Y$K$4
```

However, the real purpose of **FROM_ACCESS** is to permit more complex and subtle alterations, as shown in the example below. The **authrewrite** keyword alone is appropriate if you want to add a **Sender:** header line (showing the SMTP AUTH authenticated submitter address) to incoming messages. However, suppose you want to force the addition of such a **Sender:** header line to incoming messages only if the SMTP AUTH authenticated submitter address differs from the envelope **From:** address (that is, not bother to add a **Sender:** header line if the addresses match), and suppose further that you wish the SMTP AUTH and envelope **From:** addresses will not be considered to differ merely because the envelope **From:** includes optional subaddress information.

FROM_ACCESS

```
! If no authenticated address is available, do nothing
*|SMTP*|*|tcp_auth|*| $Y
! If authenticated address matches envelope From:, do nothing
*|SMTP*|*|tcp_auth|*|$3* $Y
! If authenticated address matches envelope From: sans
! subaddress, do nothing
*|SMTP*|*|tcp_auth|*+$3*@$5* $Y
! Fall though to...
! ...authenticated address present, but didn't match, so force
! Sender: header
*|SMTP*|*|tcp_auth|*|* $Y$K$4
```

The **\$**(metacharacter in a **FROM_ACCESS** specifies that an address should be read from the result string and used to replace the current overriding postmaster address. **\$**) has the same effect with the added constraint that the overriding postmaster address must not be set prior to invoking

the mapping. This allows for specific postmaster addresses to be used with addresses in nonlocal domains - domain postmaster addresses by definition only work with locally defined domains. The override address is (currently) the last string read from the FROM_ACCESS result prior to reading any \$N/\$F failure result.

18.3.4

PORT_ACCESS Mapping Table

The Dispatcher is able to selectively accept or reject incoming connections based on IP address and port number. At Dispatcher startup time, the Dispatcher will look for a mapping table named PORT_ACCESS. If present, the Dispatcher will format connection information in the following form:

TCP|server-address|server-port|client-address|client-port

The Dispatcher tries to match against all PORT_ACCESS mapping entries. If the result of the mapping contains \$N or \$F, the connection will be immediately closed. Any other result of the mapping indicates that the connection is to be accepted. \$N or \$F may optionally be followed by a rejection message. If present, the message will be sent back down the connection just prior to closure. Note that a CRLF terminator will be appended to the string before it is sent back down the connection.

Note – The MMP does not make use of the PORT_ACCESS mapping table. If you wish to reject SMTP connections from certain IP addresses and you are using the MMP, you must use the TCPAccess option. See “[7.5.1 To Configure Mail Access with MMP](#)” on page 187 control SMTP connections using mapping tables, use the INTERNAL_IP mapping table (see “[18.6.1 Allowing SMTP Relaying for External Sites](#)” on page 559

The flag \$< followed by an optional string causes Messaging Server to send the string to syslog (UNIX) or to the event log (NT) if the mapping probe matches. The flag \$> followed by an optional string causes Messaging Server to send the string as to syslog (UNIX) or to the event log (NT) if access is rejected. If bit 1 of the LOG_CONNECTION MTA option is set and the \$N flag is set so that the connection is rejected, then also specifying the \$T flag will cause a “T” entry to be written to the connection log. If bit 4 of the LOG_CONNECTION MTA option is set, then site-supplied text may be provided in the PORT_ACCESS entry to include in the “C” connection log entries. To specify such text, include two vertical bar characters in the right-hand side of the entry, followed by the desired text. [Table 18-3](#) lists the available flags.

In earlier versions of Messaging Server (6.2 and before) the PORT_ACCESS mapping was only reevaluated by the SMTP server (as opposed to the dispatcher) when bit 4 (value 16) of the LOG_CONNECTION MTA option was set, SMTP auth was enabled, or both. Additionally, evaluation only occurred when an AUTH, EHLO, or HELO command was issued. This has now been changed; PORT_ACCESS is now evaluated unconditionally as soon as the SMTP server thread starts, before the banner is sent.

TABLE 18-3 PORT_ACCESS Mapping Flags

Flag	Description
\$Y	Allow access.
\$U	Selectively enable channel level debugging.
Flags with arguments, in argument reading order+	
\$< string	Send string to syslog (UNIX) or to the event log (NT) if probe matches.
\$> string	Send string to syslog (UNIX) or to the event log (NT) if access is rejected.
\$N string	Reject access with the optional error text string
\$F string	Synonym for \$N string; that is, reject access with the optional error text string
Ruleset	Not used, but you must enter an empty value (double bar with no space, " ") if you want to use any of the flags below.
Realm	Not used, but you must enter an empty value (double bar with no space, " ") if you want to use any of the flags below.
Application Info	If the LOG_CONNECTION MTA option is set to bit 4 (value 16), PORT_ACCESS is allowed to add text to application information string. This is where the string can be specified. If it is not used, you must enter an empty value (double bar with no space, " ") if you want to use any of the flags below.
\$D	Causes an additional argument to be read from the template result after the mandatory SMTP auth ruleset and realm, and optional application information addition. This value must be an integer with the same semantics as the BANNER_PURGE_DELAY value. That is, it specifies the number of centiseconds to delay before purging and sending the banner. A value of 0 disabled both the delay and purge. Note that any PORT_ACCESS mapping setting overrides the BANNER_PURGE_DELAY SMTP channel option. See “14.10.1 Anti-Spam Technique: Delay Sending the SMTP Banner” on page 510 for details on using this anti-spam feature.
\$T text	If bit 1 (value 2) of the LOG_CONNECTION MTA option is set and the \$N flag is set so that the connection is rejected, then \$T outputs the entire right hand side text in a "T" record. The T log entry will include the entire mapping result string (\$N and its string). In contrast, bit 4 of LOG_CONNECTION is a different effect: it will cause material after two vertical bars to be included in normal "C" (connection close) records.

+To use multiple flags with arguments, separate the arguments with the vertical bar character, |, placing the arguments in the order listed in this table.

For example, the following mapping will only accept SMTP connections (to port 25, the normal SMTP port) from a single network, except for a particular host singled out for rejection without explanatory text:

PORT_ACCESS

```
TCP|*|25|192.123.10.70|* $N500
TCP|*|25|192.123.10.*|* $Y
TCP|*|25|*|* $N500$ Bzzzt$ thank$ you$ for$ playing.
```

Note that you will need to restart the Dispatcher after making any changes to the PORT_ACCESS mapping table so that the Dispatcher will see the changes. (If you are using a compiled MTA configuration, you will first need to recompile your configuration to get the change incorporated into the compiled configuration.)

The PORT_ACCESS mapping table is specifically intended for performing IP-based rejections. For more general control at the email address level, the SEND_ACCESS or MAIL_ACCESS mapping table, might be more appropriate.

18.3.5

IP_ACCESS Mapping Table

The IP_ACCESS Mapping Table can be used to do a last moment check on the IP address to which the MTA is about to connect; the connection attempt can then be aborted or redirected. This can be useful under certain special circumstances, for example, security concerns about a destination IP address to which should never be connected, or where it is wished to avoid connecting to known-to-be-bogus destination IP addresses (for example, 127.0.0.1), or where you wish to attempt to fail over to another destination IP address similar to a `lastresort` keyword effect (see “[12.4.3.7 Last Resort Host](#)” on page 374).

This access mapping is consulted during SMTP client operations just prior to attempting to open connections to a remote server. The mapping probe has the following format:

```
source-channel|address-current|address-count|ip-current|hostname
```

`source-channel` is the channel from which the message is being dequeued. `address-count` is the total number of IP addresses for the remote server. `address-current` is the index of the current IP address being tried. `ip-current` is the current IP address. `hostname` is the symbolic name of the remote server. The table below shows the flags for this table.

TABLE 18-4 IP_ACCESS Mapping Table Flags

Flag	Description
\$N	Immediately reject the message with an "invalid host/domain error." Any supplied text will be logged as the reason for rejection but will not be included in the DSN.
\$I	Skip the current IP without attempting to connect.
\$A	Replace the current IP address with the mapping result.

18.3.6 To Limit Specified IP Address Connections to the MTA

To limit how often a particular IP address can connect to the MTA, see [Chapter 19, “Throttling Incoming Connections Using MeterMaid.”](#) Limiting connections by particular IP addresses can be useful for preventing excessive connections used in denial-of-service attacks. In the past, this function was performed using the shared library, `conn_throttle.so` in the Port Access mapping table. No new enhancements are planned for `conn_throttle.so` and MeterMaid is its more effective replacement.

`conn_throttle.so` is a shared library used in a `PORT_ACCESS` mapping table to limit MTA connections made too frequently from particular IP addresses. All configuration options are specified as parameters to the connection throttle shared library as follows:

```
$[msg-svr-base/lib/conn_throttle.so,throttle,IP-address,max-rate]
```

IP-address is the dotted-decimal address of the remote system. *max-rate* is the connections per minute that shall be the enforced maximum rate for this IP-address.

The routine name `throttle_p` may be used instead of `throttle` for a penalizing version of the routine. `throttle_p` will deny connections in the future if they've connected too many times in the past. If the maximum rate is 100, and 250 connections have been attempted in the past minute, not only will the remote site be blocked after the first 100 connections in that minute, but they'll also be blocked during the second minute. In other words, after each minute, `max-rate` is deducted from the total number of connections attempted and the remote system is blocked as long as the total number of connections is greater than the maximum rate.

If the IP-address specified has not exceeded the maximum connections per minute rate, the shared library callout will fail.

If the rate has been exceeded, the callout will succeed, but will return nothing. This is done in a `$C/$E` combination as in the example:

```
PORt_ACCESS
TCP|*|25|*|* \
$C$[msg-svr-base/lib/conn_throttle.so,throttle,$1,10] \
$N421$ Connection$ not$ accepted$ at$ this$ time$E
```

Where,

`$C` continues the mapping process starting with the next table entry; uses the output string of this entry as the new input string for the mapping process.

`$[msg-svr-base/lib/conn_throttle.so,throttle,$1,10]` is the library call with `throttle` as the library routine, `$1` as the server IP Address, and `10` the connections per minute threshold.

`$N421$ Connection$ not$ accepted$ at$ this$ time` rejects access and returns the `421` SMTP code (transient negative completion) along with the message “Connection not accepted at this time.”

\$E ends the mapping process now. It uses the output string from this entry as the final result of the mapping process.

18.4 When Access Controls Are Applied

Messaging Server checks access control mappings as early as possible. Exactly when this happens depends upon the email protocol in use—when the information that must be checked becomes available.

For the SMTP protocol, a FROM_ACCESS rejection occurs in response to the MAIL FROM: command, before the sending side can send the recipient information or the message data. A SEND_ACCESS or MAIL_ACCESS rejection occurs in response to the RCPT TO: command, before the sending side gets to send the message data. If an SMTP message is rejected, Messaging Server never accepts or sees the message data, thus minimizing the overhead of performing such rejections.

If multiple access control mapping tables exist, Messaging Server checks them all. That is, a FROM_ACCESS, a SEND_ACCESS, an ORIG_SEND_ACCESS, a MAIL_ACCESS, and ORIG_MAIL_ACCESS mapping tables may all be in effect.

PORT_ACCESS is called from dispatcher as soon as it accepts the incoming TCP connection. It is also called from `tcp_smtp_server` when any of the `maysaslserver` or `mustsaslserver` keywords are present on the source channel. (See “[12.4.4 SMTP Authentication, SASL, and TLS](#)” on page 376.)

FROM_ACCESS is used by the `tcp_smtp_server` when processing the MAIL FROM SMTP command.

SEND_ACCESS and ORIG_SEND_ACCESS tables are used by the `tcp_smtp_server` when processing the RCPT TO SMTP command.

MAIL_ACCESS and ORIG_MAIL_ACCESS tables are used by the `tcp_smtp_server` when processing the RCPT TO SMTP command.

18.5 To Test Access Control Mappings

The `imsimta test -rewrite` utility—particularly with the `-from`, `-source_channel`, `-sender` and `-destination_channel` options—can be useful in testing access control mappings.

See “[imsimta test](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for details. The example below shows a sample SEND_ACCESS mapping table and the resulting probe.

MAPPING TABLE:

SEND_ACCESS

```
tcp_local|friendly@siroe.com|l|User@sesta.com      $Y
tcp_local|unwelcome@varrius.com|l|User@sesta.com  $NGo$ away!
```

PROBE:

```
$ TEST/REWRITE/FROM="friendly@siroe.com" -
_ $ /SOURCE=tcp_local/DESTINATION=l User@sesta.com
...
Submitted address list:
l
User (SESTA.COM) *NOTIFY FAILURES* *NOTIFY DELAYS* Submitted
notifications list:

$ TEST/REWRITE/FROM="unwelcome@varrius.com" -
_ $ /SOURCE=tcp_local/DESTINATION=l User@sesta.com
...
Submitted address list:
Address list error -- 5.7.1 Go away! User@sesta.com

Submitted notifications list:
```

18.6 To Add SMTP Relaying

Messaging Server is, by default, configured to block attempted SMTP relays; that is, it rejects attempted message submissions to external addresses from unauthenticated external sources (external systems are any other system than the host on which the server itself resides). This default configuration is quite aggressive in blocking SMTP relaying in that it considers all other systems to be external systems.

IMAP and POP clients that attempt to submit messages via the Messaging Server system's SMTP server destined for external addresses, and who do not authenticate using SMTP AUTH (SASL), will find their submission attempts rejected. Thus, you will likely want to modify your configuration so that it recognizes your own internal systems and subnets from which relaying should always be accepted.

Which systems and subnets are recognized as internal is normally controlled by the **INTERNAL_IP** mapping table, which may be found in *msg-svr-base/config/mappings*

For instance, on a Messaging Server whose IP address is 123.45.67.89, the default **INTERNAL_IP** mapping table would appear as follows:

INTERNAL_IP

```
$(123.45.67.89/32)    $Y
```

```
127.0.0.1      $Y  
*   $N
```

Here the initial entry, using the \$(IP-pattern/significant-prefix-bits) syntax, is specifying that any IP address that matches all 32 bits of 123.45.67.89 should match and be considered internal. The second entry recognizes the loopback IP address 127.0.0.1 as internal. The final entry specifies that all other IP addresses should not be considered internal. Note that all entries must be preceded by at least one space.

You may add additional entries by specifying additional IP addresses or subnets before the final \$N entry. These entries must specify an IP address or subnet (using the \$(.../...) syntax to specify a subnet) on the left side and \$Y on the right side. Or you may modify the existing \$(.../...) entry to accept a more general subnet.

For instance, if this same sample site has a class-C network, that is, it owns all of the 123.45.67.0 subnet, then the site would want to modify the initial entry by changing the number of bits used in matching the address. In the mapping table below, we change from 32 bits to 24 bits. This allows all clients on the class-C network to relay mail through this SMTP relay server.

INTERNAL_IP

```
$(123.45.67.89/24)  $Y  
127.0.0.1  $Y  
*   $N
```

Or if the site owns only those IP addresses in the range 123.45.67.80-123.45.67.99, then the site would want to use:

INTERNAL_IP

```
! Match IP addresses in the range 123.45.67.80-123.45.67.95  
$(123.45.67.80/28) $Y  
! Match IP addresses in the range 123.45.67.96-123.45.67.99  
$(123.45.67.96/30) $Y  
127.0.0.1 $Y  
* $N
```

Note that the **imsimta test -match** utility can be useful for checking whether an IP address matches a particular \$(.../...) test condition. The **imsimta test -mapping** utility can be more generally useful in checking that your INTERNAL_IP mapping table returns the desired results for various IP address inputs.

After modifying your INTERNAL_IP mapping table, be sure to issue the **imsimta restart** command (if you are not running with a compiled configuration) or an **imsimta cnbuild** followed by an **imsimta restart smtp** (if you are running with a compiled configuration) so that the changes take effect.

Further information on the mapping file and general mapping table format, as well as information on `imsimta` command line utilities, can be found in the Messaging Server Reference Manual.

18.6.1

Allowing SMTP Relaying for External Sites

All internal IP addresses should be added to the `INTERNAL_IP` mapping table as discussed above. If you have friendly or companion systems/sites from which you wish to allow SMTP relaying, the simplest approach is to include them along with your true internal IP addresses in your `INTERNAL_IP` mapping table.

If you don't wish to consider these as true internal systems/sites, (for instance, if for logging or other control purposes you wish to distinguish between *true internal systems* versus the *friendly non-internal systems with relay privileges*), there are other ways to configure the system.

One approach is to set up a special channel for receiving messages from such friendly systems. Do this by creating a `tcp_friendly` channel akin to your existing `tcp_internal` channel with official host name `tcp_friendly-daemon`, and a `FRIENDLY_IP` mapping table akin to your `INTERNAL_IP` mapping table that lists the friendly system IP addresses. Then right after the current rewrite rule:

```
! Do mapping lookup for internal IP addresses
[]    $E$R${INTERNAL_IP,$L}$U%[$L]@tcp_intranet-daemon
```

add a new rewrite rule:

```
! Do mapping lookup for "friendly", non-internal IP addresses
[]    $E$R${FRIENDLY_IP,$L}$U%[$L]@tcp_friendly-daemon
```

An alternate approach is to add to your `ORIG_SEND_ACCESS` mapping table above the final `$N` entry, new entries of the form

```
tcp_local|*@siroe.com|tcp_local|*      $Y
```

where `siroe.com` is the name of a friendly domain, and to add an `ORIG_MAIL_ACCESS` mapping table of the form:

```
ORIG_MAIL_ACCESS
```

```
TCP|*|25|$(match-siroe.com-IP-addresses)|*|SMTP*|MAIL|  \
tcp_local|*@siroe.com|tcp_local|*      $Y
TCP|*|*|*|*|SMTP*|MAIL|tcp_local|*|tcp_local|* $N
```

where the `$(...)` IP address syntax is the same syntax described in the previous section. The `ORIG_SEND_ACCESS` check will succeed as long as the address is ok, so we can go ahead and also do the `ORIG_MAIL_ACCESS` check which is more stringent and will only succeed if the IP address also corresponds to an `siroe.com` IP address.

18.7 Configuring SMTP Relay Blocking

You can use access control mappings to prevent people from relaying SMTP mail through your Messaging Server system. For example, you can prevent people from using your mail system to relay junk mail to hundreds or thousands of Internet mailboxes.

By default, Messaging Server prevents all SMTP relaying activity, including relaying by local POP and IMAP users.

Blocking unauthorized relaying while allowing it for legitimate local users requires configuring Messaging Server to know how to distinguish between the two classes of users. For example, local users using POP or IMAP depend upon Messaging Server to act as an SMTP relay.

To prevent SMTP relay, you must be able to:

- Differentiate Between Internal and External Mail
- “[18.7.2 Differentiate Authenticated Users' Mail](#)” on page 562
- “[18.7.3 Prevent Mail Relay](#)” on page 562

To enable SMTP relay by internal hosts and clients, you must add your “internal” IP addresses or subnets to the INTERNAL_IP mapping table.

18.7.1

How the MTA Differentiates Between Internal and External Mail

In order to block mail relaying activities, the MTA must first be able to differentiate between internal mail originated at your site and external mail originated out on the Internet and passing through your system back out to the Internet. The former class of mail you want to permit; the latter class you want to block. This differentiation is achieved using the switchchannel keyword on your inbound SMTP channel, usually the tcp_local channel, and is set by default.

The switchchannel keyword works by causing the SMTP server to look at the actual IP address associated with the incoming SMTP connection. Messaging Server uses that IP address, in conjunction with your rewrite rules, to differentiate between an SMTP connection originated within your domain and a connection from outside of your domain. This information can then be used to segregate the message traffic between internal and external traffic.

The MTA configuration described below is setup by default so that the server can differentiate between your internal and external message traffic.

- In the configuration file, immediately before the local channel, is a defaults channel with the noswitchchannel keyword:

```

! final rewrite rules
defaults noswitchchannel
! Local store
ims-ms ...

```

- The incoming TCP/IP channel specifies the `switchchannel` and `remotehost` keywords; for example:

```

tcp_local smtp single_sys mx switchchannel remotehost
TCP-DAEMON

```

- After the incoming TCP/IP channel definition, is a similar channel with a different name; for example:

```

tcp_intranet smtp single_sys mx allowswitchchannel routelocal
tcp_intranet-daemon

```

The `routelocal` channel keyword causes the MTA, when rewriting an address to the channel, to attempt to “short circuit” any explicit routing in the address through this channel, thereby blocking possible attempts to relay by means of looping through internal SMTP hosts via explicitly source routed addresses.

With the above configuration settings, SMTP mail generated within your domain will come in via the `tcp_intranet` channel. All other SMTP mail will come in via the `tcp_local` channel. Mail is distinguished between internal and external based upon which channel it comes in on.

How does this work? The key is the `switchchannel` keyword. The keyword is applied to the `tcp_local` channel. When a message comes in your SMTP server, that keyword causes the server to look at the source IP address associated with the incoming connection. The server attempts a reverse-pointing envelope rewrite of the literal IP address of the incoming connection, looking for an associated channel. If the source IP address matches an IP address or subnet in your `INTERNAL_IP` mapping table, the rewrite rule which calls out to that mapping table causes the address to rewrite to the `tcp_intranet` channel.

Since the `tcp_intranet` channel is marked with the `allowswitchchannel` keyword, the message is switched to the `tcp_intranet` channel and comes in on that channel. If the message comes in from a system whose IP address is not in the `INTERNAL_IP` mapping table, the reverse-pointing envelope rewrite will either rewrite to the `tcp_local` or, perhaps to some other channel. However, it will not rewrite to the `tcp_intranet` channel and since all other channels are marked `noswitchchannel` by default, the message will not switch to another channel and will remain with the `tcp_local` channel.

Note – Note that any mapping table or conversion file entries which use the string “`tcp_local`” may need to be changed to either “`tcp_*`” or “`tcp_intranet`” depending upon the usage.

18.7.2

Differentiate Authenticated Users' Mail

Your site might have “local” client users who are not part of your physical network. When these users submit mail, the message submissions come in from an external IP address—for instance, arbitrary Internet Service Providers. If your users use mail clients that can perform SASL authentication, then their authenticated connections can be distinguished from arbitrary other external connections. The authenticated submissions you can then permit, while denying non-authenticated relay submission attempts. Differentiating between authenticated and non-authenticated connections is achieved using the `saslswitchchannel` keyword on your inbound SMTP channel, usually the `tcp_local` channel.

The `saslswitchchannel` keyword takes an argument specifying the channel to switch to; if an SMTP sender succeeds in authenticating, then their submitted messages are considered to come in the specified switched to channel.

▼ To Add Distinguishing Authenticated Submissions

- 1 In your configuration file, add a new TCP/IP channel definition with a distinct name; for example:

```
tcp_auth smtp single_sys mx mustsaslserver noswitchchannel TCP-INTERNAL
```

This channel should not allow regular channel switching (that is, it should have `noswitchchannel` on it either explicitly or implied by a prior `defaults` line). This channel should have `mustsaslserver` on it.

- 2 Modify your `tcp_local` channel by adding `maysaslserver` and `saslswitchchannel` `tcp_auth`, as shown in the following example:

```
tcp_local smtp mx single_sys maysaslserver saslswitchchannel \
tcp_auth switchchannel
|TCP-DAEMON
```

With this configuration, SMTP mail sent by users who can authenticate with a local password will now come in the `tcp_auth` channel. Unauthenticated SMTP mail sent from internal hosts will still come in `tcp_internal`. All other SMTP mail will come in `tcp_local`.

18.7.3

Prevent Mail Relay

Now to the point of this example: preventing unauthorized people from relaying SMTP mail through your system. First, keep in mind that you want to allow local users to relay SMTP mail. For instance, POP and IMAP users rely upon using Messaging Server to send their mail. Note that local users may either be physically local, in which case their messages come in from an internal IP address, or may be physically remote but able to authenticate themselves as local users.

You want to prevent random people out on the Internet from using your server as a relay. With the configuration described in the following sections, you can differentiate between these classes of users and block the correct class. Specifically, you want to block mail from coming in your `tcp_local` channel and going back out that same channel. To that end, an `ORIG_SEND_ACCESS` mapping table is used.

An `ORIG_SEND_ACCESS` mapping table may be used to block traffic based upon the source and destination channel. In this case, traffic from and back to the `tcp_local` channel is to be blocked. This is realized with the following `ORIG_SEND_ACCESS` mapping table:

`ORIG_SEND_ACCESS`

```
tcp_local|*|tcp_local|*      $NRelaying$ not$ permitted
```

In this example, the entry states that messages cannot come in the `tcp_local` channel and go right back out it. That is, this entry disallows external mail from coming in your SMTP server and being relayed right back out to the Internet.

An `ORIG_SEND_ACCESS` mapping table is used rather than a `SEND_ACCESS` mapping table so that the blocking will not apply to addresses that originally match the `ims-ms` channel (but which may expand via an alias or mailing list definition back to an external address). With a `SEND_ACCESS` mapping table one would have to go to extra lengths to allow outsiders to send to mailing lists that expand back out to external users, or to send to users who forward their messages back out to external addresses.

18.7.4 To Use DNS Lookups Including RBL Checking for SMTP Relay Blocking

In the Messaging Server, there are a number of different ways to ensure that all mail accepted for delivery or forwarding comes from an address with a valid DNS name. The simplest way is to put the `mailfromdnsverify` channel keyword on the `tcp_local` channel.

Messaging Server also provides the `dns_verify` program which allows you to ensure that all mail accepted for delivery or forwarding comes from an address with a valid DNS name using the following rule in `ORIG_MAIL_ACCESS`:

`ORIG_MAIL_ACCESS`

```
TCP|*|*|*|*|SMTP*|MAIL|*|*@*|*|*  \
$[msg-svr-base/lib/dns_verify.so,\n
dns_verify,$7|$y|$$NInvalid$ host:$ $$7$ -$ %e]
```

The line breaks in the above example are syntactically significant in such mapping entries. The backslash character is a way of legally continuing on to the next line.

The `dns_verify` image can also be used to check incoming connections against things like the RBL (Realtime Blackhole List), MAPS (Mail Abuse Prevention System, DUL (Dial-up User List), or ORBS (Open Relay Behavior-modification System) lists as another attempt to protect against UBE. As with the new `mailfromdnsverify` keyword, there's also a separate "simpler to configure" approach one can use for such checks rather than doing the `dns_verify` callout. The simpler approach is to use the `DNS_VERIFY_DOMAIN` option in the `dispatcher.cnf` file. For example, in the [SERVICE=SMTP] section, set instances of the option to the various lists you want to check against:

```
[SERVICE=SMTP]
PORT=25
! ...rest of normal options...
DNS_VERIFY_DOMAIN=sbl-xbl.spamhaus.org.
DNS_VERIFY_DOMAIN=list.dsbl.org.
...etc...
```

In this case, messages are rejected at the SMTP level, that is, the messages are rejected during the SMTP dialogue and thus never sent to the MTA. The disadvantage of this simpler approach is that it does the checks for all normal incoming SMTP messages including those from internal users. This is less efficient and potentially problematic if your Internet connectivity goes down. An alternative is to call out to `dns_verify` from a `PORT_ACCESS` mapping table or `ORIG_MAIL_ACCESS` mapping table. In the `PORT_ACCESS` mapping table, you can have an initial entry or entries that don't check for local internal IP addresses or message submitters and a later entry that does the desired check for everyone else. Or, in an `ORIG_MAIL_ACCESS` mapping table, if you only apply the check on messages coming in the `tcp_local` channel then you're skipping it for messages coming from your internal systems/clients. Examples using the entry points to `dns_verify` are shown below.

PORT_ACCESS

```
! Allow internal connections in unconditionally
*|*|*|* $C$|INTERNAL_IP;$3|$Y$E
! Check other connections against RBL list
TCP|*|25|*|* \
$C$[<msg-svr-base/lib/dns_verify.so,dns_verify_domain_port,$1,\n
dnsblock.siroe.com,Your$ host$ ($1)$ found$ on$ dnsblock$ list]$E
* $YEXTERNAL
```

ORIG_MAIL_ACCESS

```
TCP|*|25|*|*|SMTP*|*|tcp_local|*@|*|* \
$C$[msg-svr-base/lib/dns_verify.so,\n
dns_verify_domain,$1,sbl-xbl.spamhaus.org.]$E
```

For more information see: [Performance Tuning Realtime BlockLists \(RBL\) Lookups \(<http://wikis.sun.com/display/CommSuite/Performance+tuning+DNS+Realtime+BlockLists+%28RBL%29+lookups>\).](http://wikis.sun.com/display/CommSuite/Performance+tuning+DNS+Realtime+BlockLists+%28RBL%29+lookups)

The PORT_ACCESS table is probed both by the dispatcher, when accepting connections, and by the `tcp_smtp_server` process under certain circumstances.

`tcp_smtp_server` processes always check the PORT_ACCESS table for channels marked `maysaslserver` or `mustsaslserver`, and they will do it for all channels if bit 4 (value 16) of the LOG_CONNECTION in option.dat is set.

So if either of those are true, the customer needs to be aware that his PORT_ACCESS table is being processed twice for every connection. This may be a trivial overhead except that callouts to things like DNS RBLs or LDAP lookups can be relatively expensive in terms of their impact on SMTP server response time. For this reason, you want to avoid doing them any more than necessary. That is one reason the callout in the example above is coded after the INTERNAL_IP lookup. If the SMTP client is in your INTERNAL_IP table, then you allow the connection without doing the RBL lookup. Reversing that order would mean your local clients would be delayed by RBL lookups. To prevent doing this twice, add a check for one the flags set by dispatcher or `tcp_smtp_server` so that you only process that table entry when one of those is set or not. For example, add \$:A to the entry:

```
TCP|*|25|*|*  \
$C$:A$[<msg-svr-base/lib/dns_verify.so,dns_verify_domain_port,$1,\n
dnsblock.siroe.com,Your$ host$ ($1)$ found$ on$ dnsblock$ list]$E
* $YEXTERNAL
```

We added the \$:A after the \$C so the table processing will continue down the table if this does not match. The \$:A specifies that this entry be processed only if it is being done by the dispatcher, that is, not when done by `tcp_smtp_server`. Alternatively, if you wanted to cause it to be done only in `tcp_smtp_server`, check for S instead:

```
TCP|*|25|*|*  \
$C$:S$[<msg-svr-base/lib/dns_verify.so,dns_verify_domain_port,$1,\n
dnsblock.siroe.com,Your$ host$ ($1)$ found$ on$ dnsblock$ list]$E
* $YEXTERNAL
```

The negative check would be \$:A to check that A is not set, or \$:S to check that S is not set.

18.7.4.1

Support for DNS-based Databases

The `dns_verify` program supports DNS-based databases used to determine incoming SMTP connections that might send unsolicited bulk mail. Some of the publicly available DNS databases do not contain TXT records that are typically used for this purpose. Instead, they only contain A records.

In a typical setup, the TXT record found in the DNS for a particular IP address contains an error message suitable to return to the SMTP client when refusing a message. But, if a TXT record is not found and an A record is found, then versions of `dns_verify` prior to Messaging Server 5.2 returned the message “*No error text available.*”

`dns_verify` now supports an option that specifies a default text that is used in the event that no `TXT` record is available. For example, the following `PORT_ACCESS` mapping table shows how to enable this option:

```
PORT_ACCESS

*|*|*|*|* $C$|INTERNAL_IP;$3|$Y$E \
TCP|*|25|*|* \
$C$[<msg-srv-base/lib/dns_verify.so \
,dns_verify_domain_port,$1,dnsblock.siroe.com,Your$ host$ ($1)$ \
found$ on$ dnsblock$ list]$E
* $YEXTERNAL
```

In this example, if the remote system is found in a query in the domain `dnsblock.siroe.com`, but no `TXT` record is available, then the following message is returned, “*Your host a.b.c.d found on dnsblock list.*”

18.8 Handling Large Numbers of Access Entries

Sites that use very large numbers of entries in mapping tables should consider organizing their mapping tables to have a few general wildcarded entries that call out to the general text database for the specific lookups. It is much more efficient to have a few mapping table entries calling out to the general text database for specific lookups than to have huge numbers of entries directly in the mapping table.

One case in particular is that some sites like to have per user controls on who can send and receive Internet email. Such controls are conveniently implemented using an access mapping table such as `ORIG_SEND_ACCESS`. For such uses, efficiency and performance can be greatly improved by storing the bulk of the specific information (e.g., specific addresses) in the general text database with mapping table entries structured to call out appropriately to the general text database.

For example, consider the `ORIG_SEND_ACCESS` mapping table shown below.

```
ORIG_SEND_ACCESS

! Users allowed to send to Internet
!
*|adam@siroe.com|tcp_local|*    $Y
*|betty@siroe.com|tcp_local|*    $Y
! ...etc...
!
! Users not allowed to send to Internet
!
*|norman@siroe.com|tcp_local|*  $NInternet$ access$ not$ permitted
```

```

*|opal@siroe.com|tcp_local|*      $NInternet$ access$ not$ permitted
! ...etc...
!
! Users allowed to receive from the Internet
!
tcp_*|*|*|adam@siroe.com          $Y
tcp_*|*|*|betty@siroe.com         $Y
! ...etc...
!
! Users not allowed to receive from the Internet
!
tcp_*|*|*|norman@siroe.com       $NInternet$ e-mail$ not$ accepted
tcp_*|*|*|opal@siroe.com         $NInternet$ e-mail$ not$ accepted
! ...etc...

```

Rather than using such a mapping table with each user individually entered into the table, a more efficient setup (much more efficient if hundreds or thousands of user entries are involved) is shown in the example below, which shows sample source text file for a general database and a sample ORIG_SEND_ACCESS mapping table. See “[10.9.1 MTA Text Databases](#)” on page [265](#) for set up information.

DATABASE ENTRIES

```

SEND|adam@domain.com    $Y
SEND|betty@domain.com   $Y
! ...etc...
SEND|norman@domain.com  $NInternet$ access$ not$ permitted
SEND|opal@domain.com   $NInternet$ access$ not$ permitted
! ...etc...
RECV|adam@domain.com    $Y
RECV|betty@domain.com   $Y
! ...etc...
RECV|norman@domain.com  $NInternet$ e-mail$ not$ accepted
RECV|opal@domain.com   $NInternet$ e-mail$ not$ accepted

```

MAPPING TABLE

ORIG_SEND_ACCESS

```

! Check if may send to Internet
!
*|*|*|tcp_local           $C${SEND|$1}$E
!
! Check if may receive from Internet
!
tcp_*|*|*|*                $C${RECV|$3}$E

```

In this example, the use of the arbitrary strings SEND| and RECV| in the general database left-hand sides (and hence in the general database probes generated by the mapping table) provides a way to distinguish between the two sorts of probes being made. The wrapping of the general text database probes with the \$C and \$E flags, as shown, is typical of mapping table callouts to the general text database.

The above example showed a case of simple mapping table probes getting checked against general text database entries. Mapping tables with much more complex probes can also benefit from use of the general text database.

18.9 PART 2. MAILBOX FILTERS

Mailbox filters, also called Sieve filters, filter messages containing specified strings found in the message headers and apply specified actions to these mail message. Administrators can filter mail streams going to a user, through a channel, or through an MTA. Messaging Server filters are stored on the server and evaluated by the server, hence, they are sometimes called server-side rules (SSR).

This part contains the following sections:

- [“18.9 PART 2. MAILBOX FILTERS” on page 568](#)
- [“18.10 Sieve Filter Support” on page 568](#)
- [“18.11 Sieve Filtering Overview” on page 570](#)
- [“18.12 To Create User-level Filters” on page 570](#)
- [“18.13 To Create Channel-level Filters” on page 570](#)
- [“18.14 To Create MTA-Wide Filters” on page 573](#)
- [“18.15 To Debug User-level Filters” on page 574](#)

18.10 Sieve Filter Support

Messaging Server filters are based on the Sieve filtering language, Draft 9 of the Sieve Internet Draft. See RFC3028 for more information about Sieve syntax and semantics. In addition, Messaging Server also supports the following Sieve extensions:

- **jettison.** Similar to `discard` in that it causes messages to be silently discarded, but unlike `discard`, which does nothing but cancel the implicit `keep`, `jettison` forces a `discard` to be performed. The behavioral difference is only relevant when multiple Sieve filters are involved. For example, a system level `discard` can be overridden by a user Sieve filter explicitly specifying `keep`, whereas a system level `jettison` will override anything done by a user Sieve.
- **Head-of-household Sieve filters.** Provides a means by which one user can specify a Sieve filter for another user. Uses two LDAP attributes in a user entry controlled by these MTA options:

- **LDAP_PARENTAL_CONTROLS** - Specifies an attribute containing a string value of either Yes or No. Yes means a head of household Sieve is to be applied to this entry, No means no such Sieve is to be applied. No default.
 - **LDAP_FILTER_REFERENCE** - Specifies an attribute containing a DN pointing to a directory entry where the head of household Sieve can be found. No default.
- The entry containing the head of household Sieve must contain two attributes specified by the following MTA options:
- **LDAP_HOH_FILTER** - Specifies an attribute containing the head of household Sieve. The value of this option defaults to `mailSieveRuleSource`.
 - **LDAP_HOH_OWNER** - Specifies an attribute containing the email address of the owner of the head of household. The value of this option defaults to `mail`.
- Both of these attributes must be present for the head of household Sieve to work.
- Sieve redirect can now add three header fields:

```
resent-date: date-of-resend-operation
resent-to: address-specified-in-redirect
resent-from: address-of-sieve-owner
```

The new :resent and :noresent arguments to redirect can be used to control whether or not these fields are added. If neither argument is specific the system default is used. The system default is controlled by the new `SIEVE_REDIRECT_ADD_RESENT` MTA option. Setting the option to 1 causes these fields to be generated unless :noresent used. A setting of 0 causes the fields to be generated only if :resent is used. The option defaults to 1, which means the fields are generated by default for regular redirects.

- Sieve redirect has been enhanced with three new arguments:
 - :resetmailfrom - Reset the envelope FROM: address to that of the current Sieve owner.
 - :keepmailfrom - Preserve the envelope FROM: address from the original message.
 - :notify - Specify a new set of notification flags for the redirected message. A single parameter is required giving a list of notification flags. The same set of flags accepted by the `NOTIFY` parameter of the DSN SMTP extension are accepted here: `SUCCESS`, `FAILURE`, `DELAY` and `NEVER`. Note that these flags are specified as a Sieve list, for example:

```
redirect :notify ["SUCCESS","FAILURE"] "foo@example.com";
```

The default if :notify isn't specified as the normal SMTP default of `FAILURE`, `DELAY`. :keepmailfrom is the default unless :notify is specified, in which case the default switches to :resetmailfrom. The one additional exception is that specification of the `SUCCESS` flag forces the use of :resetmailfrom unconditionally.

18.11 Sieve Filtering Overview

A Sieve filter consists of one or more conditional actions to apply to a mail message depending upon a string found in the message header. As an administrator, you can create channel-level filters and MTA-wide filters to prevent delivery of unwanted mail. Users can create per-user filters for their own mailboxes using Messenger Express. Specific instructions for this are described in the Messenger Express on-line help.

The server applies filters in the following priority:

1. User-level filters

If a personal mailbox filter explicitly accepts or rejects a message, then filter processing for that message finishes. But if the recipient user had no mailbox filter—or if the user's mailbox filter did not explicitly apply to the message in question—Messaging Server next applies the channel-level filter. Per-user filters are set

2. Channel-level filter

If the channel-level filter explicitly accepts or rejects a message, then filter processing for that message finishes. Otherwise, Messaging Server next applies the MTA-wide filter, if there is one.

3. MTA-wide filter

By default, each user has no mailbox filter. When a user uses the Messenger Express interface to create one or more filters, then their filters are stored in the Directory and retrieved by the MTA during the directory synchronization process.

18.12 To Create User-level Filters

Per-user mail filters apply to messages destined for a particular user's mailbox. Per-user mail filters can only be created using Messenger Express.

18.13 To Create Channel-level Filters

Channel-level filters apply to each message enqueued to a channel. A typical use for this type of filter is to block messages going through a specific channel.

TABLE 18-5 filter Channel Keyword URL-pattern Substitution Tags (Case-insensitive)

Tag	Meaning
*	Perform group expansion.

TABLE 18-5 filter Channel Keyword URL-pattern Substitution Tags (Case-insensitive) (Continued)

Tag	Meaning
**	Expand the attribute <code>mailForwardingAddress</code> . This can be a multivalued attribute resulting in several delivery addresses being produced.
\$\$	Substitute in the \$ character
\$\	Force subsequent text to lower case
\$^	Force subsequent text to upper case
\$_	Perform no case conversion on subsequent text
\$~	Substitute in the file path for the home directory associated with the local part of the address
\$1S	As \$S, but if no subaddress is available just insert nothing
\$2S	As \$S, but if no subaddress is available insert nothing and delete the preceding character
\$3S	As \$S, but if no subaddress is available insert nothing and ignore the following character
\$A	Substitute in the address, local-part@ host.domain
\$D	Substitute in host.domain
\$E	Insert the value of the second spare attribute, <code>LDAP_SPARE_1</code>
\$F	Insert the name of the delivery file (<code>mailDeliveryFileURL</code> attribute)
\$G	Insert the value of the second spare attribute, <code>LDAP_SPARE_2</code>
\$H	Substitute in host
\$I	Insert the hosted domain (part of UID to the right of the separator specified by <code>domainUidSeparator</code>). Fail if no hosted domain is available
\$1I	As \$I, but if no hosted domain is available just insert nothing
\$2I	As \$I, but if no hosted domain is available insert nothing and delete the preceding character
\$3I	As \$I, but if no hosted domain is available insert nothing and ignore the following character
\$L	Substitute in local-part
\$M	Insert the UID, stripped of any hosted domain
\$P	Insert the method name (<code>mailProgramDeliveryInfo</code> attribute)
\$S	Insert the subaddress associated with the current address. The subaddress is that part of the user part of the original address after the subaddress separator, usually +, but can be specified by the MTA option <code>SUBADDRESS_CHAR</code> . Fail if no subaddress is given
\$U	Insert the mailbox part of the current address. This is either the whole of the address to the left of the @ sign, or that part of the left hand side of the address before the subaddress separator, +.

▼ To Create a Channel-level Filter

- 1 Write the filter using Sieve.
- 2 Store the filter in a file in the following directory:

msg-svr-base/config/file.filter

The file must be world readable and owned by the MTA's uid.

- 3 Include the following in the channel configuration:

`destinationfilter file:IMTA_TABLE:file.filter`

- 4 Recompile the configuration and restart the Dispatcher.

Note that changes to the filter file do not require a recompile or restart of the Dispatcher.

The `destinationfilter` channel keyword enables message filtering on messages enqueued *to* the channel to which it is applied. The `sourcefilter` channel keyword enables message filtering on messages enqueued *by* (from) the channel to which it is applied. These keywords each have one required parameter which specifies the path to the corresponding channel filter file associated with the channel.

The syntax for the `destinationfilter` channel keyword is:

`destinationfilter URL-pattern`

`sourcefilter URL-pattern` where *URL-pattern* is a URL specifying the path to the filter file for the channel in question. In the following example, *channel-name* is the name of the channel.

`destinationfilter file:///usr/tmp/filters/channel-name.filter`

The `filter` channel keyword enables message filtering on the channels to which it is applied. The keyword has one required parameter which specifies the path to the filter files associated with each envelope recipient who receives mail via the channel.

The syntax for the `filter` channel keyword is

`filter URL-pattern`

URL-pattern is a URL that, after processing special substitution sequences, yields the path to the filter file for a given recipient address. *URL-pattern* can contain special substitution sequences that, when encountered, are replaced with strings derived from the recipient address, *local-part@host.domain* in question. These substitution sequences are shown in [Table 18–5](#).

The `fileinto` keyword specifies how to alter an address when a mailbox filter `fileinto` operator is applied. The following example specifies that the folder name should be inserted as a subaddress into the original address, replacing any originally present subaddress:

```
fileinto $U+$S@$D
```

18.14 To Create MTA-Wide Filters

MTA-wide filters apply to all messages enqueued to the MTA. A typical use for this type of filter is to block unsolicited bulk email or other unwanted messages regardless of the messages' destinations. To create an MTA-wide filter:

▼ To Create MTA-Wide Filters

- 1 **Write the filter using Sieve**
- 2 **Store the filter in the following file:**
`msg-srv-base/config/imta.filter`

This filter file must be world readable. It is used automatically, if it exists.

- 3 **Recompile the configuration and restart the Dispatcher**

When using a compiled configuration, the MTA-wide filter file is incorporated into the compiled configuration.

18.14.1 Routing Discarded Messages Out the FILTER_DISCARD Channel

By default, messages discarded via a mailbox filter are immediately discarded (deleted) from the system. However, when users are first setting up mailbox filters (and perhaps making mistakes), or for debugging purposes, it can be useful to have the deletion operation delayed for a period.

To have mailbox filter discarded messages temporarily retained on the system for later deletion, first add a `filter_discard` channel to your MTA configuration with the `notices` channel keyword specifying the length of time (normally number of days) to retain the messages before deleting them, as shown in the following example:

```
filter_discard notices 7
FILTER-DISCARD
```

Then set the option FILTER_DISCARD=2 in the MTA option file. Messages in the filter_discard queue area should be considered to be in an extension of users' personal wastebasket folders. As such, note that warning messages are never sent for messages in the filter_discard queue area, nor are such messages returned to their senders when a bounce or return is requested. Rather, the only action taken for such messages is to eventually silently delete them, either when the final notices value expires, or if a manual bounce is requested using a utility such as imsimta return.

Prior to Messaging Server 6 2004Q2, the use of the filter_discard channel by the jettison Sieve action was controlled by the FILTER_DISCARD MTA option. This is now controlled by the option FILTER_JETTISON, which takes its default from the FILTER_DISCARD setting. FILTER_DISCARD in turn defaults to 1 (discards go to the bitbucket channel).

18.15 To Debug User-level Filters

If a user complains that a Sieve filter is not behaving as expected, there are a number of steps you can take to debug the filters. These are described here.

▼ To Debug User-level Filters

- 1 For fileinto filtering to work, check that the ims-ms channel in the imta.cnf file is marked as follows:

```
fileinto $U+$S@$D
```

- 2 Get the user level filters from the user's LDAP entry.

User level filters are stored in their LDAP entry under the MailSieveRuleSource attribute(s). To retrieve this with a ldapsearch command, remember they are base64 encoded so you will need to decode the output by using the -Bo switch.

```
./ldapsearch -D "cn=directory manager" -w password -b  
"o=alcatraz.sesta.com,o=isp" -Bo uid=test
```

The imsimta test -rewrite command, described below, will also automatically decode them.

- 3 Verify that the user's filters are being seen by the MTA.

Issue the command:

```
# imsimta test -rewrite -filter -debug user@sesta.com
```

This should output the user's Sieve filters that you retrieve in the previous step. If you do not see the filters, then you need to figure out why the LDAP entry isn't returning them. If the imsimta test -rewrite output shows the filters, then you know that the user's filters are being seen by the MTA. The next step is to test the interpretation of the filters by using the imsimta test -expression command.

- 4 Use `imsimta test -exp` to debug the user's filter. The following information is required:**
 - a. The user's Sieve language statements from their `mailSieveRuleSource` attribute. See the steps above.**
 - b. The rfc2822 message that was supposed to trigger the filter.**
 - c. Description of what they expected the filter to do to the message.**
- 5 Create a text file (example: `temp.filter`) containing the Sieve language statements based on the user's `mailSieveRuleSource: values`. Example:**

```
require "fileinto";
if anyof(header :contains
["To","Cc","Bcc","Resent-to","Resent-cc",
"Resent-bcc"] "commsqa"){
    fileinto "QMSG";
}
```

Expected result: if commsqa is a recipient for this message, then file the message into a folder called QMSG.

- 6 Create a text file called `test.msg` that contains the contents of the rfc2822 message file supplied by user.**

You can either use a `.msg` file from the user's message store area, or create a text file called `test_rfc2822.msg` that contains the contents of the rfc2822 message file supplied by user.

- 7 Use the `imsimta test -exp` command:**

```
# imsimta test -exp -mm -block -input=temp.filter -message=test_rfc2822.msg
```

- 8 Examine the output.**

The last lines of the `imsimta test -exp` command will show the result of the Sieve interpretation. They will look like this:

```
Sieve Result: []
or this:
Sieve Result: [action]
```

where `action` is the action that would be done as a result of applying the Sieve filter on this message.

If the criteria of the filter matched, you will have some action displayed as the result. If nothing matched, then the Sieve result will be blank, and there is either a logic error in the Sieve filter or the `.msg` file doesn't contain matching information. If you get any other error, then there is a syntax error in the Sieve script file, and you need to debug it.

For more details on the output, see “[18.15.1 imsimta test -exp Output](#)” on page 576.

- 9 If the filter is syntactically valid and results are correct, then the next step is to examine a `tcp_local_slave.log` debug log file.

It may be that the message file you're testing and the one being sent aren't identical. The only way to see what's being received is to examine a `tcp_local_slave.log` file. This log will show you the exact message being sent to the MTA and how the filter is being applied to that message.

For more information on getting a `tcp_local_slave.log` debug file, see the `slave_debug` keyword in “[12.11.2 Debugging Keywords](#)” on page 417.

18.15.1 imsmta test -exp Output

The full command `imsmta test -exp` for is as follows:

```
# imsmta test -exp -mm -block -input=temp.filter -message=rfc2822.msg
```

An example of the output is as follows:

EXAMPLE 18-4 imsmta test -exp Output

```
# imsmta test -exp -mm -block -input tmp.filter -message=rfc2822.msg
Expression: if header :contains ["to"] ["pamw"]          (1)
Expression: {
Expression: redirect "usr3@sesta.com";
Expression: keep;
Expression: }
Expression:
Expression: Dump: header:2000114;0 3 1 :contains 1 "to" 1
"pamw" if 8 ;
Dump: redirect:2000121;0 1 1 "usr3@sesta.com" ; keep:2000117;0 (2)
Dump: 0
Result: 0
Filter result: [ redirect "usr3@sesta.com" keep ]      (3)
```

1) The Expression: output lines show the filter being read and parsed from `tmp.filter` text file. These are not particularly useful in debugging the script.

2) The Dump: output lines are the result of the computer interpreting the Sieve statements. You should not see any errors and the output should seem to match your input. For example the dump shows the word `redirect, usr3@sesta.com` which is like the line in the filter file `redirect "usr3@sesta.com";`

If it didn't show this matching text, then you'd be concerned, otherwise, these also are not particularly useful in debugging the script.

3) At the bottom of the output you will get the Filter result: statement. As stated earlier there are two possible results:

Sieve Result: [] or this: Sieve Result: [action]

where action is the action taken by the Sieve script. Note that sometimes the results are expected to be empty. For example, for a discard filter, you should test that it doesn't always discard every .msg file you test it against. If there is some action between the brackets, for example:

Filter result: [fileinto "QMSG" keep]

This means the text in the rfc2822.msg file matched the filter criteria. In this particular example, the filter will file the mail into folder QMSG and keep a copy in the inbox. The resulting actions in this case are fileinto and keep.

When testing filters you should test various .msg files for both results. You should always test that messages that match your filter are filtered, and messages that you do not want to match are not filtered.

Keep in mind that if for wildcard matches, you must use the :matches test and not :contains. For example, if you wish from=*[@sesta.com to match, you must use :matches or the test will fail as it will not ever satisfy the test condition.

18.15.2 imsimta test -exp Syntax

`imsimta test -exp` tests Sieve language statements against a specified RFC2822 message and sends the results of the filter to standard output.

The syntax is as follows:

```
imsimta test -exp -mm -block -input=Sieve_language_scriptfile
-mESSAGE=rfc2822_message_file
```

where,

`-block` treats the entire input as a single Sieve script. The default is to treat each line as a separate script and to evaluate it separately. The Sieve will only be evaluated once the end of file is reached.

`-input=Sieve_file` is a file containing the Sieve script. The default is to read the test script lines or script block from stdin.

`-MESSAGE=message_file` is a text file containing the RFC 2822 message you want to test your Sieve script against. This has to be an RFC 2822 message only. It cannot be a queue file (not a zz*.00 file).

Once activated, this command reads script information, evaluates it in the context of the test message, and writes out the result. The result shows what actions would be taken as well as the result of evaluating the final statement in the script.

Additional useful qualifiers are:

-from=*address* specifies the envelope from: address to be used in envelope tests. The default is to use the value specified by the RETURN_ADDRESS MTA option.

-output=*file* writes results to *file*. The default is to write the results of script evaluation to stdout.

Throttling Incoming Connections Using MeterMaid

MeterMaid is a server that can provide centralized metering and management of connections and transactions, including through monitoring IP addresses SMTP envelope addresses. Functionally, MeterMaid can be used to limit how often a particular IP address can connect to the MTA. Limiting connections by particular IP addresses is useful for preventing excessive connections used in denial-of-service attacks. MeterMaid supplants `conn_throttle.so` by providing similar functionality, but extending it across the Messaging Server installation. No new enhancements are planned for `conn_throttle.so` and MeterMaid is its more effective replacement.

This section consists of the following subsections:

- “[19.1 Technical Overview](#)” on page 579
- “[19.2 Theory of Operations](#)” on page 580
- “[19.3 Configutil Parameters for MeterMaid](#)” on page 580
- “[19.4 Limit Excessive IP Address Connections Using Metermaid—Example](#)” on page 583

19.1 Technical Overview

`conn_throttle.so` is a shared library used as a callout from the MTA’s mapping table that uses an in-memory table of incoming connections to determine when a particular IP address has recently connected too often and should be turned away for awhile. While having an in-memory table is good for performance, its largest cost is that each individual process on each server maintains its own table.

In most cases, the `conn_throttle.so` callout is done in the `PORT_ACCESS` mapping that is accessed by the Dispatcher, a single process on each system. The only cost is that there is a separate table per server.

The primary improvement by MeterMaid is that it maintains a single repository of the throttling information that can be accessed by all systems and processes within the Messaging Server environment. It continues to maintain an in-memory database to store this data to

maximize performance. Restarting MeterMaid will lose all information previously stored, but since the data is typically very short lived, the cost of such a restart (done infrequently) is very low.

19.2 Theory of Operations

MeterMaid's configuration is stored in `msg.conf` and is maintained by `configutil`.

MeterMaid is accessed from the MTA through a mapping table callout using `check_metermaid.so`. It can be called from any of the `*_ACCESS` tables. When called from the `PORT_ACCESS` table, it can be used to check limits based on the IP address of the connection which will be the most common way to implement MeterMaid as a replacement for the older `conn_throttle.so`. If called from other `*_ACCESS` tables, MeterMaid can also be used to establish limits on other data such as the envelope from or envelope to addresses as well as IP addresses.

Only one entry point in `check_metermaid.so` is defined. The `throttle` routine contacts MeterMaid providing two subsequent arguments separated by commas. The first is the name of the table against which the data will be checked, and the second is the data to be checked.

If the result from the probe is that the particular data being checked has exceeded its quota in that table, `check_metermaid.so` returns "success" so that the mapping engine will continue processing this entry. The remainder of the entry would then be used to handle this connection that has exceeded its quota.

`PORt_ACCESS`

```
*|*|*|*|* $C$|INTERNAL_IP;$3|$Y$E  
*|*|*|*|* $C$:$A$[/opt/SUNWmsgsr/lib/check_metermaid.so,throttle,tablename,$3]$N421$ \  
Connection$ declined$ at$ this$ time$E  
* $YEXTERNAL
```

Note the `$:A` flag test in the mapping table entry before the call to `check_metermaid.so`. This is to ensure that we only do the MeterMaid probe when `PORt_ACCESS` is being checked by the dispatcher as it will set the `A` flag for its probe.

19.3 Configutil Parameters for MeterMaid

MeterMaid's configuration is stored in `msg.conf` and is maintained by `configutil`. Here are some of the settings currently supported by MeterMaid. Defaults are in parenthesis. See "["configutil Parameters" in Sun Java System Messaging Server 6.3 Administration Reference](#)" for complete list of MeterMaid parameters.

Chapter 19, “Throttling Incoming Connections Using MeterMaid”

- `local.metermaid.enable` This setting must be set to yes on the system that will run the MeterMaid daemon so that the Watcher will start and control MeterMaid.
- `logfile.metermaid.*` These settings are the same as those used by imap, pop, and other services. By default MeterMaid writes its log file into `msg-svr-base/data/log/metermaid`
- `metermaid.config.listenaddr (INADDR_ANY)` The address to which MeterMaid should bind. On most systems, the default would not need to be changed, but for multi-homed or HA systems, specifying the appropriate address here is recommended.
- `metermaid.config.maxthreads (20)` The MeterMaid server is multithreaded and maintains a pool of threads onto which its tasks are scheduled. This value sets the maximum number of threads that will be used by MeterMaid. On systems with more than 4 CPUs, increasing this value may increase overall throughput.
- `metermaid.config.port (63837)` This is the port to which MeterMaid listens for connections and to which MeterMaid clients will connect.
- `metermaid.config.secret (No default; value must be supplied)` In order to authenticate incoming connections, MeterMaid uses a shared secret that the clients send once they connect to MeterMaid.
- `metermaid.config.serverhost (No default; value must be supplied)` This is the host name or IP address to which the clients will connect. It may be the same as `metermaid.config.listenaddr` but will most likely have a particular value to direct clients to one system in particular in the Messaging Server environment.

These settings are used by the `check_metermaid` client:

- `metermaid.mtaclient.connectfrequency (15)` Attempt a connection every `connectfrequency` seconds. When the client needs to connect to MeterMaid, it uses this as an internal throttle to prevent constant connection attempts when MeterMaid isn't available. During the time that the client is unable to communicate with MeterMaid, it will return a "fail" status to the MTA mapping engine indicating that MeterMaid has not blocked this connection.

For example, if `check_metermaid.so` attempts to connect to MeterMaid, but it fails for some reason, during the next N seconds as specified by `metermaid.mtaclient.connectfrequency`, no additional attempts will be attempted. It prevents `check_metermaid.so` from trying to connect to MeterMaid too frequently if it is not working.

- `metermaid.mtaclient.connectwait (5)` When the client is waiting for a connection to MeterMaid (either an initial connection or to reuse another already established connection), it will wait for `connectwait` seconds before returning a fail status and allowing this connection to continue.
- `metermaid.mtaclient.debug (no)` If this option is enabled, debugging information from the client will be printed into either the server or thread-specific log file for the SMTP server.

- `metermaid.mtaclient.maxconns` (3) In order to support multithreaded servers, the client can maintain a pool of connections to MeterMaid. By doing this, there can be increased concurrency during communications. However, due to internal locking done by MeterMaid, access to a particular table is limited to one request at a time, so multiple connections from a single process may provide limited benefit.
- `metermaid.mtaclient.readwait` (10) When communicating with MeterMaid, the client will wait `readwait` seconds before returning a fail status and allowing this connection to continue.

Lastly, the throttling tables are also defined in `msg.conf` as shown here. The * in each configuration parameter is the name of the particular table being defined. For example, for a table called `internal`, the first parameter would be called `metermaid.table.internal.data_type`.

- `metermaid.table.*.data_type` (string) MeterMaid can support two kinds of data in its tables, string and ipv4. string data is limited to 255 bytes per entry and can be compared using case-sensitive or case-insensitive functions (see `metermaid.table.*.options` below).
- `metermaid.table.*.max_entries` (1000) When MeterMaid initializes each table, it pre-allocates this many entries. MeterMaid automatically recycles old entries, even if they haven't yet expired. When a new connection is received, MeterMaid will reuse the least recently accessed entry. A site should specify a value high enough to cache the connections received during `quota_time`.
- `metermaid.table.*.options` is a comma-separated list of keywords that defines behavior or characteristics for the table. Valid keywords are:
 - `nocase` — When working with the data, all comparisons are done using a case-insensitive comparison function. (This option is valid only for string data.)
 - `penalize` — After `quota_time` seconds, throttle will normally reset the connection count to 0, but if the `penalize` option is enabled, throttle will decrement the connection count by `quota` (but not less than 0) so that additional connection attempts will penalize future `quota_time` periods. For example, if `quota` were 5 with a `quota_time` of 60, and the system received 12 connection attempts during the first minute, the first 5 connections would be accepted and the remaining 7 would be declined. After 60 seconds has passed, the number of connections counted against the particular address would be reduced to 7, still keeping it above `quota` and declining connection attempts. Assuming no additional connection attempts were made, after another 60 second period, the number of connections would be further reduced down to 2, and MeterMaid would permit connection attempts again.
- `metermaid.table.*.quota` (100) When a connection is received, it is counted against `quota`. If the number of connections received in `quota_time` seconds exceeds this value, MeterMaid will decline the connection. (The actual effect on the incoming connection is controlled by the mapping table and could result in additional scrutiny, a delay, or denying the connection.)

- `metermaid.table.*.quota_time` (60) This specifies the number of seconds during which connections will be counted against quota. After this many seconds, the number of connections counted against the incoming address will be reduced depending on the type of this table.
- `metermaid.table.*storage` (hash) MeterMaid can use two different storage methods, hash and splay. The default hash table method is recommended, but under some circumstances a splay tree may provide faster lookups.
- `metermaid.table.*.type` (throttle) Currently, the only table type supported by MeterMaid is throttle. This type of table keeps track of the data, typically IP addresses, and will throttle the incoming connections to quota connections during a period of `quota_time` seconds.

Note – Metermaid was coded to be as efficient as possible, but may not match your expectations as it uses weighted averages to limit the connection rate rather than maintaining a list of just how many connections there have been over the previous `quota_time` period.

For example, let's say you are trying to limit connections to 1250 mails per hour (`metermaid.table.tcp_auth_msg_throttle.quota = 1250`). This limits connections to an average rate of 1250/hour. In other words, if all 1250 connections are used up in the first second, then half an hour later, an additional 625 are provided. After another 15 minutes, you'd get another 362, and so on.

A common misconception is that one hour after the initial connection attempt, all will be forgiven. But Metermaid looks at an average connection rate rather than a specific count. The net effect is the same in the long run (over a 24 hour period, a total of $1250 * 24$ would be allowed), but it's not mathematically precise in a short period.

A future goals for Metermaid is that instead of offering an average throughput rate, it would maintain a distinct list of connection attempts, and be able to expire each attempt after the precise time. This has not yet been done because it's a much more computationally expensive operation to maintain than computing an average rate.

19.4 Limit Excessive IP Address Connections Using Metermaid—Example

This example uses MeterMaid to throttle IP addresses at 10 connections/minute. For reference, the equivalent `conn_throttle.so` setup in the mappings file would be as follows:

```

PORT_ACCESS
  *|*|*|*|*  $C$|INTERNAL_IP;$3|$Y$E
  *|*|*|*|*  $C$[/opt/SUNWmsgsr/lib/conn_throttle.so,throttle,$3,10]\\
$N421$ Connection$ declined$ at$ this$ time$E
  *           $YEXTERNAL

```

This PORT_ACCESS mapping table implements conn_throttle.so to restrict connections to a rate of no more than 10 connections per minute for non-INTERNAL connections.

One fundamental difference between the two technologies is that instead of configuring details such as the rate-limit for throttling directly into the mapping table, MeterMaid uses configutil parameters for these settings. This example is described below.

1. **Designate one of your systems to be the MeterMaid server host.**

On this system, set the following configutil parameter:

```
local.metermaid.enable -v TRUE
```

Set an authentication password used to verify communications between the client and MeterMaid server:

```
configutil -o metermaid.config.secret -v password
```

2. **Define a throttling table.**

MeterMaid's throttling behavior is determined by the use of named throttling tables that define operating characteristics. To define a table that throttles at a rate of 10 connections per minute, set the following parameters:

```
configutil -o metermaid.table.ext_throttle.data_type -v ipv4  
configutil -o metermaid.table.ext_throttle.quota -v 10
```

ext_throttle is the name of the throttling table. *ipv4* is the data type Internet Protocol version 4 address representation. 10 is the quota (connection limit).

3. **On the MeterMaid system, start MeterMaid.**

```
# start-msg metermaid
```

4. **On systems where the MTA will use MeterMaid to do throttling, specify the MeterMaid host and password.**

These are required:

```
configutil -o metermaid.config.secret -v MeterMaid_Password  
configutil -o metermaid.config.serverhost -v name_or_ipaddress_of_MetermaidHost
```

5. **Set up the MeterMaid PORT_ACCESS table.**

This table is similar to the equivalent conn_throttle.so setup:

PORT_ACCESS

```
*|*|*|*|* $C$|INTERNAL_IP;$3|$Y$E  
*|*|*|*|* $C$:$A$[/opt/SUNWmsgsr/lib/check_metermaid.so,throttle,\  
ext_throttle,$3] $N421$ Connection$ declined$ at$ this$ time$E  
* $YEXTERNAL
```

The first line checks to see if the IP address attempting a connection is internal. If it is, it allows the connection. The second line runs the IP address through MeterMaid and if it has connected too frequently, it declines the connection. The third line allows any other connections through, but flagged as EXTERNAL.

Note that this call to `check_metermaid.so` is very similar to the callout to `conn_throttle.so`. The function in `check_metermaid.so` is the same. `throttle` and its arguments are simply the table name as configured using `metermaid.table.tablename` and the IP address to check (\$3). Like `conn_throttle.so`, this function returns `success` when the limit (as specified in `metermaid.table.ext_throttle.quota`) has been reached. This allows the remainder of the mapping entry line is processed, which sends a message (421 SMTP code, transient negative completion, *Connection not accepted at this time*) to the remote SMTP client, and tells the Dispatcher to close the connection.

Note also that `$:A` ensures that this line will only be processed when being called from the Dispatcher. Without this, the call to `check_metermaid.so` would also happen in the context of the `tcp_smtp_server` processes which also probes the `PORT_ACCESS` mapping table. This would cause MeterMaid to count each incoming connection twice.

This is the basic configuration to set up MeterMaid as a `conn_throttle.so` replacement. See “[10.3.2 Mapping Operations](#)” on page 241 and “[18.3.4 PORT_ACCESS Mapping Table](#)” on page 552 for information on these topics.

19.4.1

Additional Useful MeterMaid Options

Two additional configuration options may be useful in some circumstances. The `conn_throttle.so` shared library also had a `throttle_p` function that would penalize connections that exceeded the limit by applying a consequence for an extended period beyond the basic 60 seconds. This same behavior is available in MeterMaid by configuring the following option on the MeterMaid server system:

```
configutil -o metermaid.table.ext_throttle.options -v penalize
```

This changes the behavior for the `ext_throttle` table so that connections may be penalized for connection attempts greater than the value set for `metermaid.table.ext_throttle.quota`.

The other option is relevant for systems that receive a large number of connections. Because MeterMaid is able to keep track of connections throughout the distributed MTA environment, it's possible that the limit of the number of connections being retained in MeterMaid's internal in-memory database may be insufficient for the overall volume of the MTA environment. The default is 1000 entries per table, but if you anticipate having more than 1000 connections per minute throughout your MTA environment, you can increase this number through this configuration option:

```
configutil -o metermaid.table.ext_throttle.max_entries -v max_entries
```

Note that even if the *max_entries* is reached during the 60 second period, MeterMaid will automatically discard the oldest and least frequently used entries. Thus, the more frequently connecting systems will remain in MeterMaid's table to be counted, keeping enough information to provide effective throttling.

Managing the Message Store

This chapter describes the message store and its administration interface. This chapter contains the following sections:

- “[20.1 Overview](#)” on page 587
- “[20.2 Message Store Directory Layout](#)” on page 589
- “[20.3 How the Message Store Removes Messages](#)” on page 593
- “[20.4 Specifying Administrator Access to the Store](#)” on page 594
- “[20.5 About Shared Folders](#)” on page 596
- “[20.6 Shared Folder Tasks](#)” on page 598
- “[20.7 Managing Message Types](#)” on page 606
- “[20.8 About Message Store Quotas](#)” on page 616
- “[20.9 To Set the Automatic Message Removal \(Expire and Purge\) Feature](#)” on page 625
- “[20.10 Configuring Message Store Partitions](#)” on page 635
- “[20.11 Performing Message Store Maintenance Procedures](#)” on page 638
- “[20.12 Backing Up and Restoring the Message Store](#)” on page 646
- “[20.13 Monitoring User Access](#)” on page 659
- “[20.14 Troubleshooting the Message Store](#)” on page 660
- “[20.15 Migrating or Moving Mailboxes to a New System](#)” on page 675

20.1 Overview

The message store contains the user mailboxes for a particular Messaging Server instance. The size of the message store increases as the number of mailboxes, folders, and log files increase. You can control the size of the store by specifying limits on the size of mailboxes (disk quotas), by specifying limits on the total number of messages allowed, and by setting aging policies for messages in the store.

As you add more users to your system, your disk storage requirements increase. Depending on the number of users your server supports, the message store might require one physical disk or

multiple physical disks. There are two ways to integrate this additional disk space into your system. The easiest way is to add additional message store partitions (see “[20.10 Configuring Message Store Partitions](#)” on page 635)

Likewise, if you are supporting multiple hosted domains, you might want to dedicate a server instance to a single, large domain. With this configuration, you can designate a store administrator for a particular domain. You can also expand the message store by adding more partitions.

To manage the message store, Messaging Server provides a set of command-line utilities described in [Table 20–1](#). For information about using these utilities, see “[20.11 Performing Message Store Maintenance Procedures](#)” on page 638 and the [Sun Java System Messaging Server 6.3 Administration Reference](#).

TABLE 20–1 Message Store Command-line Utilities

Utility	Description
configutil	Sets and modifies configuration parameters for the store.
deliver	Delivers mail directly to the message store accessible by IMAP or POP mail clients.
hashdir	Identifies the directory that contains the message store for a particular user.
imsconnutil	Monitors user access of the message store.
imexpire	Automatically removes messages from the message store based on administrator-specified criteria like age.
iinitquota	Reinitializes the quota limit from the LDAP directory and recalculates the disk space being used.
imsasm	Handles the saving and recovering of user mailboxes.
imsbackup	Backs up stored messages.
imsexport	Exports Messaging Server mailboxes into UNIX /var/mail format folders.
imsrestore	Restores messages that have been backed up.
imscripster	The IMAP server protocol scripting tool. Executes a command or sequence of commands.
mboxutil	Lists, creates, deletes, renames, or moves mailboxes; reports quota usage.
mkbackupdir	Creates and synchronizes the backup directory with the information in the message store.
MoveUser	Moves a user's account from one messaging server to another.

TABLE 20–1 Message Store Command-line Utilities *(Continued)*

Utility	Description
<code>imquotacheck</code>	Calculates the total mailbox size for each user in the message store and compares the size with their assigned quota. Localized versions of <code>imquotacheck</code> notification incorrectly convert the % and the \$ signs. To correct the encoding, replace every \$ with \24 and replace every % with \25 in the message file.
<code>readership</code>	Collects readership information on shared IMAP folders.
<code>reconstruct</code>	Reconstructs mailboxes that have been damaged or corrupted.
<code>stored</code>	Performs background and daily tasks, expunges, and erases messages stored on disk.

20.2 Message Store Directory Layout

Figure 20–1 shows the message store directory layout for a server instance. The message store is designed to provide fast access to mailbox contents. The store directories are described in Table 20–2.

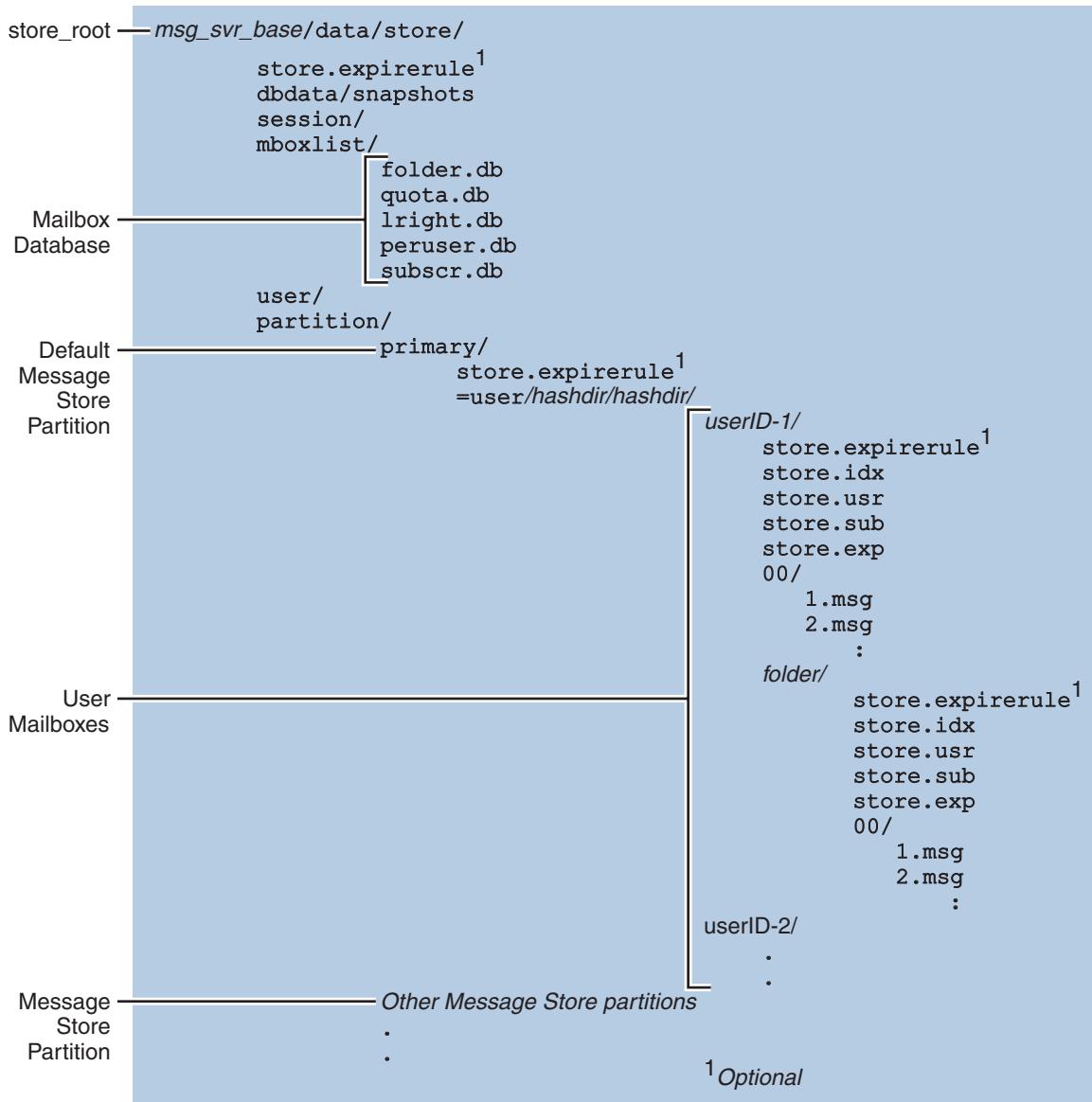


FIGURE 20-1 Message Store Directory Layout

The message store consists of a number of mailbox databases and the user mailboxes. The mailbox databases consists of information about users, mailboxes, partitions, quotas and other message store related data. The user mailboxes contain the user's messages and folders.

Mailboxes are stored in a *message store partition*, an area on a *disk* partition specifically devoted to storing the message store. See “[20.10 Configuring Message Store Partitions](#)” on page 635 for

details. Message store partitions are not the same as disk partitions, though for ease of maintenance, we recommend having one disk partition for each message store partition.

Mailboxes such as INBOX are located in the *store_root*. For example, a sample directory path might be:

store_root/partition/primary/=user/53/53/=mack1

The table below describes the message store directory.

TABLE 20-2 Message Store Directory Description

Location	Content/Description
<i>msg-svr-base</i>	Default: /opt/SUNWmsgsr The directory on the Messaging Server machine that holds the server program, configuration, maintenance, and information files.
<i>store_root</i>	<i>msg-svr-base/data/store</i> Top-level directory of the message store. Contains the <code>mboxlist</code> , <code>user</code> , and <code>partition</code> subdirectories.
<i>./store.expirerule</i>	Contains the automatic message removal rules (expire rules). This optional file can be at different locations. See “ 20.9 To Set the Automatic Message Removal (Expire and Purge) Feature ” on page 625
<i>store_root/dbdata/snapshots</i>	Message store database backup snapshots that <code>stored</code> makes periodically.
<i>store_root/mboxlist/</i>	Contains mailbox database, database (Berkeley DB) that stores information about the mailboxes and quota information. <code>folder.db</code> contains information about mailboxes, including the name of the partition where the mailbox is stored, the ACL, and a copy of some of the information in <code>store.idx</code> . There is one entry in <code>folder.db</code> per mailbox <code>quota.db</code> contains information about quotas and quota usage. There is one entry in <code>quota.db</code> per user. <code>lright.db</code> - an index for the folders by acl lookup rights. <code>peruser.db</code> contains information about per-user flags. The flags indicate whether a particular user has seen or deleted a message. <code>subscr.db</code> contains information about user subscriptions.
<i>store_root/session/</i>	Contains active message store process information.
<i>store_root/user/</i>	Not used.
<i>store_root/partition/</i>	Contains the message store partitions. A default <code>primary</code> partition is created. Place any other partitions you define in this directory.

TABLE 20–2 Message Store Directory Description *(Continued)*

Location	Content/Description
<i>store_root/partition/primary/=user/</i>	Contains all the user mailboxes in the subdirectory of the partition. The mailboxes are stored in a hash structure for fast searching. To find the directory that contains a particular user's mailbox, use the <code>hashdir</code> utility.
<i>.../=user/hashdir/hashdir/userid/</i>	The top-level mail folder for the user whose ID is <i>userid</i> . This contains the user's INBOX. For the default domain, <i>userid</i> is <i>uid</i> . For hosted domains, <i>userid</i> is <i>uid@domain</i> . A user's incoming messages are delivered to the INBOX here.
<i>.../userid/folder</i>	A user-defined mailbox on the messaging server.
<i>.../userid/store.idx</i>	An index that provides the following information about mail stored in the <i>/userid/</i> directory: number of messages, disk quota used by this mailbox, the time the mailbox was last appended, message flags, variable-length information for each message including the headers and the MIME structure, and the size of each message. The index also includes a backup copy of <code>mboxlist</code> information for each user and a backup copy of quota information for each user.
<i>.../userid/store.usr</i>	Contains a list of users who have accessed the folder. For each user listed, contains information about the last time the user accessed the folder, the list of messages the user has seen, and the list of messages the user has deleted.
<i>.../userid/store.sub</i>	Contains information about user subscriptions.
<i>.../userid/store.exp</i>	Contains a list of message files that have been expunged, but not removed from disk. This file appears only if there are expunged messages.
<i>.../userid/nn/ or .../userid/folder/nn/</i>	<i>nn</i> is a hash directory that contains messages in the format <i>message_id.msg</i> ; <i>nn</i> can be a number from 00 to 99. <i>message_id</i> is also a number. Example: messages 1 through 99 are stored in the <i>.../00</i> directory. The first message is <i>1.msg</i> , the second is <i>2.msg</i> , third <i>3.msg</i> , and so on. Messages 100 through 199 are stored in the <i>01</i> directory; messages 9990 through 9999 are stored in the <i>99</i> directory; messages 10000 through 10099 are in the <i>00</i> directory, and so on.

20.2.1 Valid UIDs and Folder Names

The following definitions will assist in this discussion:

message store user ID — A mail user's unique identifier in the message store. In the default domain, this is the same as the user's *uid* attribute in LDAP. In hosted domains this is *uid@domain* where *uid* is the *uid* LDAP attribute and *domain* is the canonical domain name.

message store mailbox name for commands — Some message store commands require that you specify a mailbox name. The required form of the name is *user userid mailbox* where *userid* is the message store user ID (see above) and *mailbox* is a user's mailbox. Specifying INBOX sometimes implies all the user's mailboxes in the message store. For example, the following command:

```
mboxutil -d user/joe/INBOX
```

will remove the INBOX and all the folders of user joe. Note that in the context of message stores, folders and mailboxes are synonymous.

Valid UIDs. Valid and invalid UID characters are controlled separately by the MTA and message store mechanisms. That means UID character limitations are specified by the union of MTA and message store limitations. The following characters and strings are invalid as UIDs in the message store:

- % ? * & / :\
- ASCII values less than 20 or greater than 7E hexidecimal (see `man ascii`)
- A leading '-' is prohibited because it is reserved for negative rights
- A leading 'group=' is prohibited because it is reserved for group IDs
- The following UIDs are reserved: 'anonymous' 'anybody' 'anyone' and 'anyone@domain'

The following characters are invalid in UIDs in the MTA:

```
<space> $~=#*+%!@,{()}/\\<>;:\\\"[]&?"
```

The list of characters forbidden by the MTA can be modified by setting the `option.dat` parameter `LDAP_UID_INVALID_CHARS` with a string of the forbidden characters using decimal ASCII values, however, you are strongly advised not to change the default constraint. The default setting is as follows and reflect the characters listed above:

```
DAP_UID_INVALID_CHARS=32,33,34,35,36,37,38,40,41,42,43,44,47,58,59,60,61,  
62,63,64,91,92,93,96,123,125,126
```

```
LDAP_UID_INVALID_CHARS=32,33,34,35,36,37,38,40,41,42,43,44,47,58,59,60,61,  
62,63,64,91,92,93,96,123,125,126
```

Valid mail folder names. The following characters are invalid as folder names:

% * ? and ASCII values less than 20 or greater than 7E hexidecimal (see `man ascii`).

In addition, folder names must be valid UTF-7 sequences.

20.3 How the Message Store Removes Messages

Messages are removed from the message store in three stages:

1. *Delete.* A client sets a message flag to *delete*. At this point, the message is marked for removal, but client can still restore the message by removing the delete flag. If there is a second client, the deleted flag may not be recognized immediately by that second client. You can set the `configutil` parameter `local imap immediateflagupdate` to enable immediate flag update.

2. *Expunge*. Messages are removed from the mailbox. Technically, they are removed from the message store index file, `store.idx`. The message itself is still on disk, but once messages are expunged, clients can no longer restore them.

Expire is a special case of expunge. Messages that conform to a set of administrator-defined removal criteria such as message size, age and so forth, are expunged. See “[20.9 To Set the Automatic Message Removal \(Expire and Purge\) Feature](#)” on page 625

3. *Purge*. The `imexpire` utility purges from the disk any messages that have been expunged at 11PM everyday by default. This can be configured with `local.schedule.expire`, which controls the message expire schedule, and `store.cleanupage`, which controls the purge grace period (period of time before which the message will not be purged). Note that this is different from the `imsmta purge` command and `configutil` parameter `local.schedule.purge` which purges older versions of the MTA log files.

20.4 Specifying Administrator Access to the Store

Message store administrators can view and monitor user mailboxes and specify access control for the message store. Store administrators have proxy authentication privileges to any service (POP, IMAP, HTTP, or SMTP), which means they can authenticate to any service using the privileges of any user. These privileges allow store administrators to run certain utilities for managing the store. For example, using `MoveUser`, store administrators can move user accounts and mailboxes from one system to another.

This section discusses how to grant store privileges to the message store for your Messaging Server installation.

Note – Other users might also have administrator privileges to the store. For example, some administrators may have these privileges.

You can perform administrator tasks as described in the following subsections:

- “[To Add an Administrator Entry](#)” on page 594
- “[To Modify an Administrator Entry](#)” on page 595
- “[To Delete an Administrator Entry](#)” on page 595
- “[20.4.1 To Protect Mailboxes from Deletion or Renaming Except by an Administrator](#)” on page 595

▼ To Add an Administrator Entry

- **Command Line:** To add an administrator entry at the command line:

```
configutil -o store.admins -v "adminlist"
```

where *adminlist* is a space-separated list of administrator IDs. If you specify more than one administrator, you must enclose the list in quotes. In addition, the administrator must be a member of the Service Administrator Group (in the LDAP user entry: `memberOf: cn=Service Administrators, ou=Groups, o=usergroup`). You must restart `imapd` for the system to recognize the change in `store.admins`.

▼ To Modify an Administrator Entry

- **Command Line.** To modify an existing entry in the message store Administrator UID list at the command line:

```
configutil -o store.admins -v "adminlist"
```

where *adminlist* is a space-separated list of administrator IDs. If you specify more than one administrator, you must enclose the list in quotes. In addition, the administrator must be a member of the Service Administrator Group (in the LDAP user entry: `memberOf: cn=Service Administrators, ou=Groups, o=usergroup`).

You must restart `imapd` for the system to recognize the change in `store.admins`.

▼ To Delete an Administrator Entry

- **Command Line.** To delete store administrators at the command line, you can edit the administrator list as follows:

```
configutil -o store.admins -v "adminlist"
```

where *adminlist* is a space-separated list of administrator IDs. If you specify more than one administrator, you must enclose the list in quotes. In addition, the administrator must be a member of the Service Administrator Group (in the LDAP user entry: `memberOf: cn=Service Administrators, ou=Groups, o=usergroup`).

You must restart `imapd` for the system to recognize the change in `store.admins`.

20.4.1

To Protect Mailboxes from Deletion or Renaming Except by an Administrator

You may wish to protect some mailboxes from deletion or modification except by the Administrator. The following procedures describes how to do this. If someone other than an Administrator attempts to delete, modify or rename a protected mailbox, the error message `mailbox is pinned` is displayed.

Set the `local.store.pin` configutil variable, using the following format:

```
configutil -o local.store.pin -v "mailbox1%" "mailbox2%" "mailbox 3"
```

where *mailbox1*, *mailbox2*, and *mailbox 3* are the mailboxes to be protected (note that spaces can be used in mailbox names), and % is the separator between each mailbox.

20.5 About Shared Folders

A *group* or *shared folder* is like any other mail folder except that other users and groups can read, delete, or add messages to it depending on the permissions given. Messages can be added to shared folders by normal drag and drop, by Sieve filters, or by sending messages directly using the form: *uid+folder@domain*.

The example below shows the address for sending email to a *private shared folder* owned by carol.fanning@siroe.com called crafts_club:

carol.fanning+crafts_club@siroe.com

This example shows the address for sending email to a *public shared folder* called tennis@siroe.com.

public+tennis@siroe.com

Shared folders are useful for starting, sharing, and archiving an ongoing email conversation on a particular topic. For example, a group of software developers can create a shared folder for discussing development of a particular project called mosaic_voices. When a message is sent or dropped into the folder mosaic_voices, anyone who has permissions to the shared folder (permissions can be added by individual addresses or by group addresses) can open this mailbox and read the message.

Shared folders are displayed in user's mailbox tree under a folder called Shared Folders. An example is shown below.

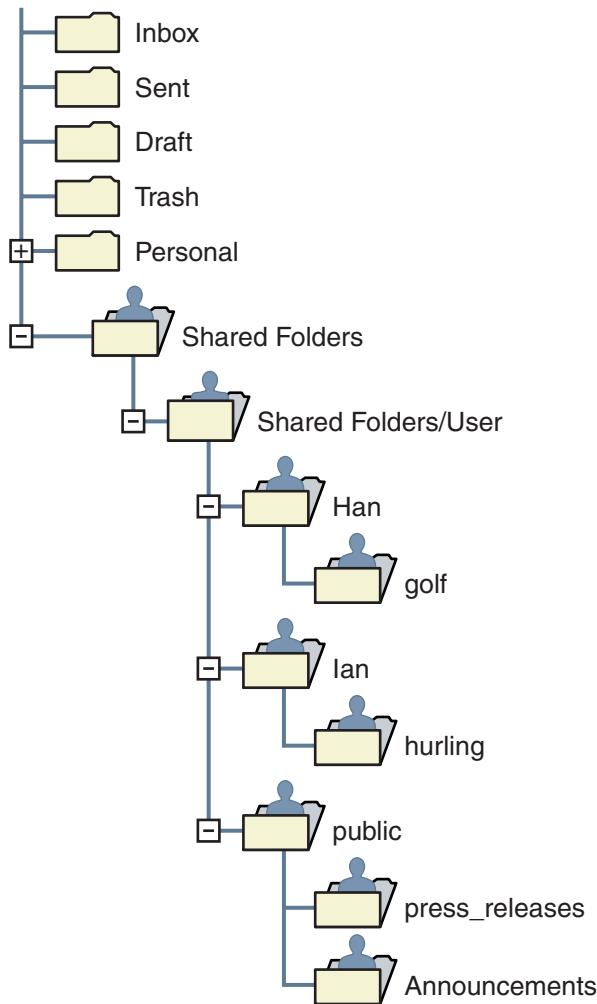


FIGURE 20–2 Example of Shared Mail Folder List as Seen from a Mail Client

There are two kinds of shared folders:

- **Private Shared Folder** – A shared folder created and owned by a specific user using Communications Express or some other mail client that supports shared folder creation. (See the Communications Express help screens for details.) Private shared folders are in the Shared Folders/User mail folder directory.
- **Public Shared Folder** – A public shared folder is created by the mail administrator, but does not have an owner. The email address of a public folder looks like this:

`public+foldername@domain`

For example, you might want a folder, such as `public+software_dev@siroe.com` for posting information about a special interest group inside the company. Interested employees would be granted access to this public folder. Public shared folders are in the Shared Folders/Public mail folder directory

Normally shared folders are only available to users on a particular message store. Messaging Server, however, allows you to create special shared folders that can be accessed across multiple message stores. These are called *distributed shared folders*. See “[20.6.4 To Set Up Distributed Shared Folders](#)” on page 603 for details.

20.6 Shared Folder Tasks

This section describes the shared folder administrator tasks:

- “[To Specify Sharing Attributes for Private Shared Folders](#)” on page 598
- “[To Create a Public Shared Folder](#)” on page 599
- “[20.6.1 To Add Shared Folders with an Email Group](#)” on page 600
- “[20.6.2 To Set or Change a Shared Folder’s Access Control Rights](#)” on page 601
- “[20.6.3 To Enable or Disable Listing of Shared Folders](#)” on page 602
- “[20.6.4 To Set Up Distributed Shared Folders](#)” on page 603
- “[20.6.5 To Monitor and Maintain Shared Folder Data](#)” on page 605

▼ To Specify Sharing Attributes for Private Shared Folders

1 Private shared folders are created by the user.

Many mail clients support the creation of private shared folders. You can try this out on Communications Express.

2 Set the sharing parameters for private shared folders.

The following configutil parameters are supported:

`store.privatesharedfolders.restrictanyone` - If enabled (1), disallow regular users from setting the permission on private shared folders to anyone. Default: 0

`store.privatesharedfolders.restrictdomain` - If enabled (1), disallow regular users sharing private folders to users outside of their domain. Default: 0

`store.privatesharedfolders.shareflags` - If 0, flags cannot be shared across users. If 1, flags can be shared across users. Default: 0

`store.publicsharedfolders.user` - Public shared folder owner's userid. Typically, this is simply `public`. Default: NULL (unset)

▼ To Create a Public Shared Folder

Public folders must be created by system administrators because they require access to the LDAP database as well as the readership command.

- 1 **Create an LDAP user entry called `public` that will act as a container for all public folders (see “[20.5 About Shared Folders](#)” on page [596](#)).**

Example:

```
dn: cn=public,ou=people,o=sesta.com,o=ISP
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: inetUser
objectClass: ipUser
objectClass: inetMailUser
objectClass: inetLocalMailRecipient
objectClass: nsManagedPerson
objectClass: userPresenceProfile
cn: public
mail: public@sesta.com
mailDeliveryOption: mailbox
mailHost: manatee.siroe.com
uid: public
inetUserStatus: active
mailUserStatus: active
mailQuota: -1
mailMsgQuota: 100
```

- 2 **Create folders within the `public` account by using the `mboxutil` command line utility.**

For example, create a public folder called `gardening`:

```
mboxutil -c user/public/gardening
```

- 3 **Set the name of the folder.**

Typically, this is `public`. Here's the command for setting the folder name to `public`:

```
configutil -o store.publicsharedfolders.user -v public
```

- 4 **Specify the users and their access rights to the shared folder.**

Use the `readership` command to specify users and their access rights. For example the following command gives everyone at `sesta.com` lookup, read, and posting access to the public folder `gardening`:

```
readership -s user/public/gardening anyone@sesta.com lrp
```

For detailed instructions on how to user readership, see “[20.6.2 To Set or Change a Shared Folder’s Access Control Rights](#)” on page [601](#)

20.6.1 To Add Shared Folders with an Email Group

Shared folders are typically created by adding users to a shared folder list with Communications Express, or by creating public shared folders as described earlier. Sometimes, however, users may wish to add an email group (mail distribution list) to a shared folder list so that everyone in the group will have access to the shared folder. For example, a group called `tennis@sesta.com` has 25 members and the members have decided that they would like to create a shared folder to store all email going to this group address.

▼ To Add an Email Group to a Shared Folder

Adding an email group to a shared folder requires System Administrator privileges.

1 Create a folder. (If this has already been done, then skip this step.)

Typically this should be done by one of the members of the group. If it’s not, you can create it for them using the following command:

```
mboxutil -c user/gregk/gardening
```

`gregk` is the uid of the shared folder owner. `gardening` is the name of the shared folder.

2 Add the attribute-value pair `aclGroupAddr group_name@domain` to the user entry of every member who will have access to the group shared folder.

Using the example above, add the following attribute-value pair to each user entry receiving access to the shared folder:

```
aclGroupAddr: tennis@sesta.com
```

Note that members will already have this attribute if the group was created dynamically using the `memberURL` attribute in the group entry. URL value for this attribute would look like this:

```
memberURL: ldap:///o=sesta.com??sub?(&(aclGroupAddr=tennis@sesta.com)  
(objectclass=inetmailuser))
```

(The sample entry line has been wrapped for typographic reasons. The actual entry should appear on one physical line.)

3 Specify the group and the access rights to the shared folder.

Use the `readership` command to do this. Using the example above the following command gives members of `tennis@sesta.com` lookup, read, and posting access to the public folder `gardening`:

```
readership -s user/gregk/gardening tennis@sesta.com lrp
```

For detailed instructions on how to user readership, see “[20.6.2 To Set or Change a Shared Folder’s Access Control Rights](#)” on page 601

20.6.2 To Set or Change a Shared Folder’s Access Control Rights

Users can set or change the access control for a shared folder using the Communications Express interface. Administrators can set or change the access control for a shared folder using the readership command line utility. The command has the following form:

```
readership -s foldername identifier rights_chars
```

where *foldername* is the name of the public folder for which you are setting rights, *identifier* is the person or group to whom you are assigning the rights, and *rights_chars* are the rights you are assigning. For the meaning of each character, see [Table 20–3](#).

Note – anyone is a special identifier. The access rights for anyone apply to all users. Similarly, the access rights for anyone@*domain* apply to all users in the same domain.

TABLE 20–3 ACL Rights Characters

Character	Description
l	lookup– User can see and subscribe to the shared folder. (IMAP commands allowed: LIST and LSUB)
r	read– Users can read the shared folder. (IMAP commands allowed: SELECT, CHECK, FETCH, PARTIAL, SEARCH, COPY from the folder)
s	seen– Directs the system to keep seen information across sessions. (Set IMAP STORE SEEN flag)
w	write– Users can mark as read, and delete messages. (Set IMAP STORE flags, other than SEEN and DELETED)
i	insert– Users can copy and move email from one folder to another. (IMAP commands allowed: APPEND, COPY into folder)
p	post– Users can send mail to the shared folder email address. (No IMAP command needed)
c	create– Users can create new sub-folders. (IMAP command allowed: CREATE)
d	delete– Users can delete entries from the shared folder. (IMAP commands allowed: EXPUNGE, set STORE DELETED flag)
a	administer– Users have administrative privileges. (IMAP command allowed: SETACL)

20.6.2.1

Examples

If you wish everyone at the `sesta` domain to have lookup, read and email marking (but not posting) access to the public folder called `golftournament`, issue the command as follows:

```
readership -s User/public/golftournament anyone@sesta.lwr
```

To assign the same access to everyone on the message store issue the following:

```
readership -s User/public/golftournament anyone lwr
```

To assign lookup, read, email marking and posting rights to a group, issue the command as follows:

```
readership -s User/public/golftournament group=golf@sesta.com lwrp
```

If you want to assign administrator and posting rights for this folder to an individual, `j doe`, issue the command as follows:

```
readership -s User/public/golftournament jdoe@sesta.com lwrpa
```

To deny an individual or group access to a public folder, prefix the user id with a dash. For example, to deny lookup, read and write rights to `jsmith`, issue the command as follows:

```
readership -s User/public/golftournament -jsmith@sesta.com lwr
```

To deny an individual or group an access right, prefix the ACL rights character with a dash. For example, to deny posting rights to `jsmith`, issue the command as follows:

```
readership -s User/public/golftournament jsmith@sesta.com -p
```

Note – Posting messages to a shared folder using the `uid+folder@domain` address requires that the `p` (post) access right be used with the readership command. See “[20.6.2 To Set or Change a Shared Folder’s Access Control Rights](#)” on page 601

20.6.3

To Enable or Disable Listing of Shared Folders

The server will or will not return shared folders when responding to a LIST command depending on the setting in the configuration option `local.store.sharedfolders`. Setting the option to `off` disables it. The setting is enabled by default (set to on).

SELECT and LSUB commands are not affected by this option. The LSUB command returns every subscribed folder, including shared folders. Users can SELECT the shared folders they own or are subscribed to.

20.6.4 To Set Up Distributed Shared Folders

Normally shared folders are only available to users on a particular message store. Messaging Server, however, allows you to create *distributed shared folders* that can be accessed across multiple message stores. That is, access rights to distributed shared folders can be granted to any users within the group of message stores. Note, however, that web mail clients (HTTP access clients like Messenger Express) do not support remote shared folders access. Users can list and subscribe to the folders, but they can't view or alter the contents.

Distributed shared folders require the following:

- The message store userids must be unique across the group of message stores.
- The directory data across the deployment must be identical.

The remote message stores (that is the message stores that do not hold the shared folder) must be configured as proxy servers by setting the configuration variables listed in [Table 20–4](#).

TABLE 20–4 Variables for Configuring Distributed Shared Folders

Name	Value	Data Format
local.service.proxy.serverlist	message store server list	space-separated strings
local.service.proxy.admin	default store admin login name	string
local.service.proxy.adminpass	default store admin password	string
local.service.proxy.admin.hostname	store admin login name for a specific host	string
local.service.proxy.adminpass.hostname	store admin password for a specific host	string

20.6.4.1 Setting Up Distributed Shared Folders—Example

[Figure 20–3](#) shows a distributed folder example of three message store servers called StoreServer1, StoreServer2, and StoreServer3.

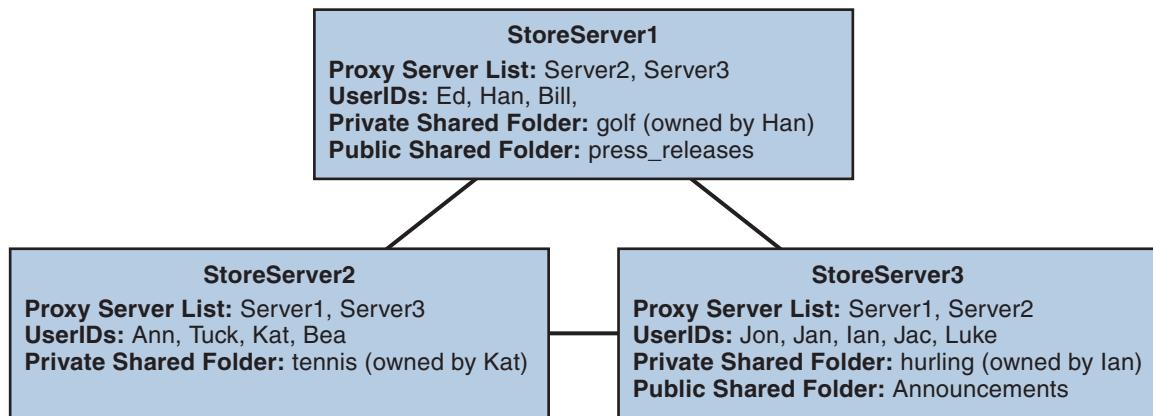


FIGURE 20-3 Distributed Shared Folders—Example

These servers are connected to each other as peer proxy message stores by setting the variables shown in [Table 20-4](#). Each server has a private shared folder—*golf* (owned by Han), *tennis* (owned by Kat), and *hurling* (owned by Luke). In addition there are two public shared folders called *press_releases* and *Announcements*. [Figure 20-2](#) shows Ed's shared folder list. Below is an example of the ACLs for each server in this configuration.

```
$ StoreServer1 :> imcheck -d lright.db
Ed: user/Han/golf
Ian: user/Han/golf
anyone: user/public/press_releases
```

```
$ StoreServer2 :> imcheck -d lright.db
Jan: user/Kat/tennis
Ann: user/Kat/tennis
anyone: user/public+Announcements user/public+press_releases
```

```
$ StoreServer3 :> imcheck -d lright.db
Tuck: user/Ian/hurling
Ed: user/Ian/hurling
Jac: user/Ian/hurling
anyone: user/public/Announcements
```

20.6.5 To Monitor and Maintain Shared Folder Data

The `readership` command line utility allows you to monitor and maintain shared folder data which is held in the `folder.db`, `peruser.db`, and `lright.db` files. `folder.db` has a record for each folder that holds a copy of the ACLs. The `peruser.db` has an entry per user and mailbox that lists the various flags settings and the last date the user accessed any folders. The `lright.db` has a list of all the users and the shared folders for which they have lookup rights.

The `readership` command line utility takes the following options:

TABLE 20-5 `readership` Options

Options	Description
<code>-d days</code>	Returns a report, per shared folder, of the number of users who have selected the folder within the specified days.
<code>-p months</code>	Removes data from the <code>peruser.db</code> for those users who have not selected their shared folders within the specified months.
<code>-l</code>	List the data in <code>lright.db</code> .
<code>-s folder_identifier_rights</code>	Set access rights for the specified folder. This updates the <code>lright.db</code> as well as the <code>folder.db</code> .

Using the various options, you can perform the following functions:

- “20.6.5.1 To Monitor Shared Folder Usage” on page 605
- “20.6.5.2 To List Users and Their Shared Folders” on page 606
- “20.6.5.3 To Remove Inactive Users” on page 606
- “20.6.5.4 To Set Access Rights” on page 606

20.6.5.1 To Monitor Shared Folder Usage

To find out how many users are actively accessing shared folders, issue the command:

```
readership -d days
```

where `days` is the number of days to check. Note that this option returns the number of active users, not a list of the active users.

Example: To find out the number of users who have selected shared folders within the last 30 days, issue the following command:

```
readership -d 30
```

20.6.5.2

To List Users and Their Shared Folders

To list users and the shared folders to which they have access, issue the command:

```
imcheck -d lright.db
```

Example output:

```
$ imcheck -d lright.db
group=lee-staff@siroe.com: user/user2/lee-staff
richb: user/golf user/user10/Drafts user/user2/lee-staff user/user10/Trash
hanl: user/public+hurling@siroe.com user/golf
gregk: user/public+hurling@siroe.com user/heaving user/tennis
```

20.6.5.3

To Remove Inactive Users

If you want to remove inactive users (those who have not accessed shared folders in a specified time period) issue the command:

```
readership -p months
```

where *months* is the number of months to check for.

Example: Remove users who have not accessed shared folders for the past six months:

```
readership -p 6
```

20.6.5.4

To Set Access Rights

You can assign access rights to a new public folder, or change access rights on a current public folder.

For an example of how to set access rights with this command, see “[20.6.2 To Set or Change a Shared Folder’s Access Control Rights](#)” on page 601

20.7 Managing Message Types

This section includes the following topics:

- “[20.7.1 Message Type Overview](#)” on page 607
- “[To Configure Message Types](#)” on page 608
- “[20.7.2 Message Types in IMAP Commands](#)” on page 610
- “[20.7.3 Sending Notification Messages for Message Types](#)” on page 612
- “[20.7.4 Administering Quotas by Message Type](#)” on page 612
- “[20.7.5 Expiring Messages by Message Type](#)” on page 614

20.7.1 Message Type Overview

A unified messaging application can receive, send, store, and administer messages of many types, including text messages, voice mail, fax mail, image data, and other data formats. The message store allows you to define up to 63 different message types.

One method of manipulating messages by type is to organize the messages by their types into individual folders.

With the introduction of the message type feature, you do not have to maintain different message types in individual mailbox folders. Once you configure a message type, the message store can identify it, no matter where it is stored. Thus, you can store heterogeneous message types in the same folder. You also can perform the following tasks:

- Track the usage of message types
- Send notifications grouped by message type
- Set and administer different quotas for different message types, whether they are stored in the same folder or different folders
- Move messages from one folder to another, according to criteria configured uniquely for each message type
- Expire messages according to criteria configured for each message type

20.7.1.1

Planning the Message-Type Configuration

In a unified messaging application, data of heterogeneous formats are given standard internet message headers so that Messaging Server can store and manage the data. For example, when voice mail is sent to an end-user's phone, a telephone front-end system adds a message header to the incoming voice mail and delivers it to the message store.

In order to recognize and administer messages of different types, all components of the unified messaging system must use the same message-type definitions and the same header fields to identify the messages.

Before you configure the message store to support message types, you must

- Plan which message types you intend to use
- Decide on the definition for each message type
- Decide which header field to use

For example, if your application includes phone messages, you can define this message type as "multipart/voice-message" and use the Content-Type header field to identify message types.

You would then configure the telephone front-end system to add the following header information to each phone message to be delivered to the message store:

Content-Type: multipart/voice-message

Next, you would configure the message store to recognize the `multipart/voice-message` message type, as described in the sections that follow.

20.7.1.2

Defining and Using Message Types

You define a message type by giving it a unique definition such as `multipart/voice-message`. By default, the message store reads the `Content-Type` header field to determine the message-type. If you prefer, you can configure a different header field to identify the message types.

The message store reads the `Content-Type` (or other specified) header field, ignoring case. That is, the message store accepts the header field as valid even if the header's combination of uppercase and lowercase letters differs from the expected combination.

The message store reads only the message-type name in the header field. It ignores additional arguments or parameters.

To define a message type, use the `configutil` utility to set values for the `store.messagetype` parameters. For instructions, see “[To Configure Message Types](#)” on page 608.

Configuring a message type allows the message store to identify and manipulate messages of the specified type. It is the first, essential step in administering message types in a unified messaging application.

To take full advantage of the message-type features provided by the message store, you also should perform some or all of the following tasks:

- Configure a JMQ notification plug-in and write Message Queue clients for retrieving notifications that track the status of the message types
- Configure quota roots that apply to each message type
- Write expire rules and set LDAP attribute values to expire and purge messages according to message type

These tasks are summarized in the following sections:

- “[20.7.3 Sending Notification Messages for Message Types](#)” on page 612
- “[20.7.4 Administering Quotas by Message Type](#)” on page 612
- “[20.7.5 Expiring Messages by Message Type](#)” on page 614

▼ To Configure Message Types

To configure a message type, use the `configutil` utility to set the `store.messagetype` parameter values that define and identify the message type.

1 Enable message types by setting the `store.messagetype.enable` parameter to on.

This `configutil` parameter allows the message store to identify and manipulate message types. You must set this parameter before you can configure an individual message type.

For example, enter the following command:

```
configutil -o store.messagetype.enable -v 1
```

2 Define and identify the message type by setting the `store.messagetype.x` parameter.

The variable *x* identifies this particular message type in the message store. The variable *x* must be an integer greater than zero and less than 64. You can define up to 63 message types by iteratively configuring this parameter with unique integers.

You define the value of the message type with a text string that describes the type.

For example, to define a text message type, you could enter the following command:

```
configutil -o store.messagetype.1 -v text/plain
```

To define a voice message type, you could enter the following command:

```
configutil -o store.messagetype.2 -v multipart/voice-message
```

3 Provide a flag name for the message type by setting the `store.messagetype.x.flagname` parameter.

This parameter creates a unique flag that identifies the message type. The flag is automatically set whenever a message of this type first arrives in the message store and remains associated with the message until it is purged. The flag name value is a text string that describes the message type. It does not have to be the same as the value set with the `store.messagetype.x` parameter.

The variable *x* is the integer ID of the message type defined with the `store.messagetype.x` parameter.

For example, to define flag names for the message types configured in the preceding step, enter the following commands:

```
configutil -o store.messagetype.1.flagname -v text
```

```
configutil -o store.messagetype.2.flagname -v voice_message
```

4 Configure a quota root name for the message type by setting the `store.messagetype.x.quotaroot` parameter.

This parameter enables the quota function to identify and manage a quota root for this message type. The parameter value is a name—a text string that describes the message type. It does not have to be the same as the value set with the `store.messagetype.x` parameter.

The variable *x* is the integer ID of the message type defined with the `store.messagetype.x` parameter.

When this parameter is configured, you can set a quota that applies to the specified message type. For more information, see “[20.7.4 Administering Quotas by Message Type](#)” on page 612.

For example, to enable the use of quota roots for the message types configured in the preceding steps, enter the following commands:

```
configutil -o store.messagetype.1.quotaroot -v text
```

```
configutil -o store.messagetype.2.quotaroot -v voice
```

5 To configure an alternate header field for identifying the message type, set the store.messagetype.header parameter.

By default, the message store reads the Content-Type header field to determine the message type. Configure the store.messagetype.header parameter only if you want to use a different header field for identifying the message type. The value of this parameter is a text string.

For example, to use a field called X-Message-Type, enter the following command:

```
configutil -o store.messagetype.header -v X-Message-Type
```

20.7.2

Message Types in IMAP Commands

When you configure the store.messagetype.x.flagname parameter for a message type, you create a unique flag that identifies the message type. This flag cannot be modified by end users.

Messaging Server presents the message-type flag as a user flag to IMAP clients. Mapping the message type to a user flag allows mail clients to use simple IMAP commands to manipulate messages by message type.

For example, you can perform the following operations:

- Use the IMAP FETCH FLAGS command to display a message-type flag name as a user-defined flag to the client.
For a sample use of the IMAP FETCH FLAGS command, see [Example 20-1](#), shown below.
- Use a message-type flag as a keyword in an IMAP SEARCH command.
For a sample use of the IMAP SEARCH command, see [Example 20-1](#), shown below.

The message-type user flag is read only. It cannot be modified by IMAP commands.

The following examples assume you configure the message-type configutil parameters with the values shown here:

```
store.messagetype.enable = yes  
  
store.messagetype.1 = text/plain  
store.messagetype.1.flagname = text  
store.messagetype.1.quotaroot = text  
  
store.messagetype.2 = multipart/voice-message  
store.messagetype.2.flagname = voice_message  
store.messagetype.2.quotaroot = voice
```

EXAMPLE 20-1 IMAP FETCH Session Based on the Message-Type configutil Configurations

The following IMAP session fetches messages for the currently selected mailbox:

```

2 fetch 1:2 (flags rfc822)
* 1 FETCH (FLAGS (\Seen text) RFC822 {164}

Date: Wed, 8 July 2006 03:39:57 -0700 (PDT)
From: bob.smith@siroe.com
To: john.doe@siroe.com
Subject: Hello
Content-Type: TEXT/plain; charset=us-ascii

* 2 FETCH (FLAGS (\Seen voice_message) RFC822 {164}

Date: Wed, 8 July 2006 04:17:22 -0700 (PDT)
From: sally.lee@siroe.com
To: john.doe@siroe.com
Subject: Our Meeting
Content-Type: MULTIPART/voice-message; ver=2.0

2 OK COMPLETED

```

In the preceding example, two messages are fetched, one text message and one voice mail.

The message-type flags are displayed in the format configured with the `store.messagetype.*.flagname` parameter.

The Content-Type header fields identify the message types. The message-type names are displayed as they were received in the incoming messages. They use mixed uppercase and lowercase letters and include the message-type arguments such as `charset=us-ascii`.

EXAMPLE 20-2 IMAP SEARCH Session Based on the Message-Type configutil Configurations

The following IMAP session searches for voice messages for the currently selected mailbox:

```

3 search keyword voice_message
* SEARCH 2 4 6
3 OK COMPLETED

```

In the preceding example, messages 2, 4, and 6 are voice messages. The keyword used in the search is `voice_message`, the value of the `store.messagetype.2.flagname` parameter.

20.7.3 Sending Notification Messages for Message Types

Notifications can deliver status information about messages of different types, such as text messages, voice mail, and image data. Messaging Server uses Sun Java System Message Queue to send notification information for message types. For information about configuring the JMQ notification plug-in for Message Queue, see [Chapter 22, “Configuring the JMQ Notification Plug-in to Produce Messages for Message Queue”](#)

To enable the JMQ notification plug-in to recognize a particular message type, you must configure the `store.messagetype` parameters, including the `store.messagetype.x.flagname` parameter. For details, see [“To Configure Message Types” on page 608](#).

Once the message types have been configured, JMQ notification messages can identify the particular message types. You can write a Message Queue client to interpret notification messages by message type and deliver status information about each type to the mail client.

The JMQ notification function counts the number of messages currently in the mailbox, by message type. Instead of sending one count, an array specifying the count for each message type is sent with the notification message.

For example, a `NewMsg` notification message can carry data to tell the user that, for example, there are seven new voice mail messages and four new text messages in the user's inbox.

For more information about sending notifications by message type, see [“22.3.3 Notifications for Particular Message Types” on page 703](#)

20.7.4 Administering Quotas by Message Type

When you set a quota for a message type, you include that value in a *quota root*. A quota root specifies quotas for a user. It can specify different quotas for particular message types and mailbox folders, and it can specify a default quota that applies to all remaining message types, folders, and messages not defined by type.

For complete information about setting and managing quotas, see [“20.8.2 Quota Theory of Operations” on page 617](#)

20.7.4.1 Before You Set Message-Type Quotas

Before you can set quotas for message types, you must configure the following parameters:

- Set the `store.messagetype.x.quotaroot` parameter for each message type. For details, see [“To Configure Message Types” on page 608](#).
- Set the `store.typequota.enable` parameter to on.

For example, enter the following command:

```
configutil -o store.typequota.enable -v 1
```

20.7.4.2

Methods of Setting Message-Type Quotas

Use one of the following methods to set quotas for message types:

- Set message-type quotas for a user with the LDAP attributes `mailQuota` or `mailMsgQuota` (or both).

For information about how to set quota roots with these attributes, see the `mailQuota` and `mailMsgQuota` entries in [Chapter 3, “Messaging Server and Calendar Server Attributes,” in *Sun Java Communications Suite 5 Schema Reference*](#).

- Set default message-type quotas that apply to all individual users when the `mailQuota` and `mailMsgQuota` attributes are not set.

To set default quotas, use the `store.defaultmessagequota` or `store.defaultmailboxquota` parameter (or both).

For information about how to set quota roots with these parameters, see “[20.8.4 Configuring Message Store Quotas](#)” on page [620](#).

When you set a quota for the message type with a `configutil` parameter or LDAP attribute shown above, you must use the quota root specified with the `store.messagetype.x.quotaroot` parameter.

20.7.4.3

Example of a Message-Type Quota Root

The example described in this section sets the following quotas for the user `joe`:

- The default mailbox storage quota is 40 M
- The default mailbox message quota is 5000
- The storage quota for the Archive folder is 100M
- The storage quota for text message types is 10 M
- The message quota for text message types is 2000
- The storage quota for voice message types is 10 M
- The message quota for voice message types is 200

This quota root permits greater storage in the Archive folder (100 M) than in all the other folders and message types combined (60 M). Also, no message limit is set for the Archive folder; in this example, only storage limits matter for archiving.

The message types have both storage and number-of-message quotas.

The message-type quotas apply to the sum of all messages of those types, whether they are stored in the Archive folder or in any other folder.

The default mailbox quotas apply to all messages that are not text or voice message types and are not stored in the Archive folder. That is, the message-type quotas and Archive quota are not counted as part of the default mailbox quotas.

To set the quota root in this example, you would take the following steps:

1. Configure the `store.messagetype.x.quotaroot` parameter as follows:

```
store.messagetype.1.quotaroot = text
```

```
store.messagetype.2.quotaroot = voice
```

2. Configure the `mailQuota` attribute for the user `joe` as follows:

```
mailQuota: 20M;#text%10M;#voice%10M;Archive%100M
```

3. Configure the `mailMsgQuota` attribute for the user `joe` as follows:

```
mailMsgQuota: 5000;#text%2000;#voice%200
```

When you run the `getquotaroot` IMAP command, the resulting IMAP session displays all quota roots for the user `joe`'s mailbox, as shown here:

```
1 getquotaroot INBOX
* QUOTAROOT INBOX user/joe user/joe/#text user/joe/#voice
* QUOTA user/joe (STORAGE 12340 20480 MESSAGE 148 5000)
* QUOTA user/joe/#text (STORAGE 1966 10240 MESSAGE 92 2000)
* QUOTA user/joe/#voice (STORAGE 7050 10240 MESSAGE 24 200)

2 getquotaroot Archive
* QUOTAROOT user/joe/Archive user/joe/#text user/joe/#voice
* QUOTA user/joe/Archive (STORAGE 35424 102400)
* QUOTA user/joe/#text (STORAGE 1966 10240 MESSAGE 92 2000)

* QUOTA user/joe/#voice (STORAGE 7050 10240 MESSAGE 24 200)
```

20.7.5

Epiring Messages by Message Type

The `expire` and `purge` feature allows you to move messages from one folder to another, archive messages, and remove messages from the message store, according to criteria you define in `expire` rules. You perform these tasks with the `imexpire` utility.

Because the `imexpire` utility is run by the administrator, it bypasses quota enforcement.

For information about how to write `expire` rules and use the `imexpire` utility, see “[20.9 To Set the Automatic Message Removal \(Expire and Purge\) Feature](#)” on page 625

You can write `expire` rules so that messages of different types are expired according to different criteria.

The `expire` feature is extremely flexible, offering many choices for setting `expire` criteria. This section describes one example in which text and voice messages are expired according to different criteria.

The example assumes you have configured text and voice message types as follows:

```
store.messagetype.1 = text/plain
store.messagetype.2 = multipart/voice-message
```

Assume also that the message store is configured to read the Content-Type header field to determine the message type.

EXAMPLE 20-3 Sample Rules for Expiring Different Message Types

```
TextInbox.folderpattern: user/%/INBOX
TextInbox.messageheader.Content-Type: text/plain
TextInbox.messagedays: 365
TextInbox.action: fileinto:Archive

VoiceInbox.folderpattern: user/%/INBOX
VoiceInbox.messageheader.Content-Type: multipart/voice-message
VoiceInbox.savedays: 14
VoiceInbox.action: fileinto:OldMail

VoiceOldMail.folderpattern: user/%/OldMail
VoiceOldMail.messageheader.Content-Type: multipart/voice-message
VoiceOldMail.savedays: 30
VoiceOldMail.action: fileinto:Trash

Trash.folderpattern: user/%/Trash
Trash.savedays: 7
Trash.action: discard
```

In this example, text messages and voice mail are expired in different ways, and they follow different schedules, as follows:

- Text messages are moved from a user's inbox to the user's Archive folder one year after they arrive in the message store.
- Voice mail is moved from the inbox to the OldMail folder after two weeks. If the user saves a voice message, the saved date is reset, and the message is moved two weeks after the new date.
- Voice mail is moved from the OldMail folder to the Trash folder after 30 days. The user also can save a voice message in the OldMail folder, which postpones the removal of the message for another 30 days after the new saved date.
- Messages of all types are discarded seven days after they are moved to the Trash folder.

The expire rules move voice mail to Trash automatically. Text messages are moved to Trash when a user deletes them.

EXAMPLE 20-3 Sample Rules for Expiring Different Message Types *(Continued)*

Note: The `savedays` rule causes a message to be expired the specified number of days after the message is saved. In a typical voice mail system, a user can save voice mail on the voice mail menu. For text messages, a message is saved when it is moved to a folder. The `messagedays` rule causes a message to be expired the specified number of days after it first arrives in the message store, no matter which folder it is stored in or how often it is moved.

20.8 About Message Store Quotas

With the explosion of email and voice mail, IMAP mailboxes can grow very large. Message store quotas limit, or *quota*, for how much disk space or how many messages can be held by a user or domain, are in a specific folder, or are of a specific message type. Quotas are used to limit or reduce message store usage. This section contains information about the following:

For further information, see “[20.11.4 To Monitor Quota Limits](#)” on page 641

- “[20.8.1 Quota Overview](#)” on page 616
- “[20.8.2 Quota Theory of Operations](#)” on page 617
- “[20.8.3 Message Store Quota Attributes and Parameters](#)” on page 618
- “[20.8.4 Configuring Message Store Quotas](#)” on page 620

20.8.1 Quota Overview

Quotas can be set for specific users or domains and can be set in terms of number of messages or number of bytes. It can also be set for specific folders and message types. Message type quotas allow you to specify limits for message type. For example, voice mail and email. Folder quotas set limits to the size of a user's folder in bytes or messages. For example, a quota can be set on the Trash folder. Messaging Server allows you to set default quotas for domains and users as well as customized quotas.

Once a quota is set, how the system responds to users or domains that are either over quota or approaching the quota is also configurable. One response is to send users an over *quota notification*. Another response is to halt delivery of messages into the message store when quota is exceeded. This is called *quota enforcement* and usually occurs after a specified *grace period*. A grace period is how long the mailbox can be over the quota before enforcement occurs. If message delivery is halted due to over quota, incoming messages remain in the MTA queue until one of the following occurs:

- The size or number of the user's messages no longer exceeds the quota, at which time the MTA delivers the messages.
- The undelivered message remains in the MTA queue longer than the specified *grace period*, at which time messages are returned to sender. (See “[20.8.4.5 To Set a Grace Period](#)” on page 624).

- The message has remained in the message queue longer than the maximum message queue time. This is controlled by the notices MTA channel keyword (see “[10.10.4.3 To Set Notification Message Delivery Intervals](#)” on page [278](#)).

For example, if your grace period is set for two days, and you exceed quota for one day, new messages continue to be received and held in the message queue, and delivery attempts continue. After the second day, the messages bounce back to the sender.

Disk space becomes available when a user deletes and expunges messages or when the server deletes messages according to expiration policies established (see “[20.9 To Set the Automatic Message Removal \(Expire and Purge\) Feature](#)” on page [625](#)).

20.8.1.1

Exceptions for Telephony Application Servers

To support unified messaging requirements, Messaging Server provides the ability to override quota limitations imposed by the message store. This guarantees the delivery of messages that have been accepted by certain agents, namely telephony application servers (TAS). Messages accepted by a TAS can be routed through a special MTA channel that ensures the message is delivered to the store regardless of quota limits. This is a fairly esoteric usage, but can be used to telephony applications. For more information about configuring an TAS channel, contact your Sun messaging representative.

Quota by message type is useful for telephony applications that use unified messaging. For example, if a mix of messages, say text and voice mail, is stored in a user's mailbox, then the administrator can set different quotas for different types of messages. The user's email can have one quota, and their voice mail can have a different quota.

20.8.2

Quota Theory of Operations

Customized user and domain quotas are specified by adding quota attributes to LDAP user and domain entries. Quota defaults, notification policy, enforcement, and grace period are specified in configutil parameters or by using the `imquotacheck` utility.

To determine if a user is over quota, Messaging Server first checks to see if a quota has been set for the individual user. If no quota has been set, Messaging Server looks at the default quota set for all users. For a user, the quota is for all the cumulative bytes or messages in all of the user's folders. For a domain, the quota is for all the cumulative bytes or messages of all the users in a particular domain. For a message type, the quota is for all the cumulative bytes or messages for that message type. For a folder, the quota is for all the cumulative bytes or messages for user's folder.

You can specify the following quota values for a user's mailbox tree:

- Quota values for specific folders in the user's mailbox.
- Quota values for specific message types such as voice mail or text messages. (A message type quota applies to messages of that type in all folders in the user's mailbox.)

- A default quota value that applies to all folders and message types in the user's mailbox that are not explicitly assigned quotas.

The following guidelines apply when you assign multiple quota values for a user:

- Quotas do not overlap. For example, when there is a quota for a particular message type or folder, messages of that type or messages in that folder are not counted toward the default quota. Each message counts toward one and only one quota.
- The total quota for the whole user mailbox equals the sum of the values of all the quotas specified by default, type, and folder.
- Message type quotas take precedence over folder quotas. For example, suppose one quota is specified for a user's memos folder and another quota is specified for voice messages. Now suppose the user stores eight voice messages in the memos folder. The eight messages are counted toward the voice mail quota and excluded from the memos folder quota.

Changes made to the quota attributes and configutil parameters will take effect automatically, but not immediately as information is stored in caches and it may take a little time before the changes fully take effect. Messaging Server provides a command, “[iminitquota](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* that updates the changed immediately.

The `imquotacheck` utility allows you to check message store usage against assigned quotas.

20.8.3 Message Store Quota Attributes and Parameters

This section lists the major the message store quota attributes and configutil parameters. The intention is to provide you with an overview of the functionality interface. For detailed information on these attributes and parameters, refer to the appropriate reference documentation.

The following table lists the quota attributes. Refer to the *Sun Java Communications Suite 5 Schema Reference*

TABLE 20–6 Message Store Quota Attributes

Attribute	Description
<code>mailQuota</code>	Bytes of disk space allowed for the user's mailbox.
<code>mailMsgQuota</code>	Maximum number of messages permitted for a user. This is a cumulative count for all folders in the store.
<code>mailUserStatus</code>	Status of the mail user. Some of the possible values are <code>active</code> , <code>inactive</code> , <code>deleted</code> , <code>hold</code> , and <code>overquota</code> .
<code>mailDomainDiskQuota</code>	Bytes of disk space allowed for the cumulative count of all the mailboxes in a domain.

TABLE 20-6 Message Store Quota Attributes *(Continued)*

Attribute	Description
<code>mailDomainMsgQuota</code>	Maximum number of messages permitted for a domain, that is, the total count for all mailboxes in the store.
<code>mailDomainStatus</code>	Status of the mail domain. Values and default are the same as <code>mailUserStatus</code> .

The following table lists the quota parameters. Refer to the [Chapter 3, “Messaging Server Configuration,” in Sun Java System Messaging Server 6.3 Administration Reference](#) for the latest and most detailed information.

TABLE 20-7 Message Store configutil parameters

Parameter	Description
<code>store.quotaenforcement</code>	Enable quota enforcement. When off, the quota database is still updated, but messages are always delivered. Default: On
<code>store.quotanotification</code>	Enable quota notification. Default: OFF
<code>store.defaultmailboxquota</code>	Store default quota by number of bytes. Default: -1 (unlimited)
<code>store.defaultmessagequota</code>	Store default quota by number of messages. Numeric. Default: -1 (unlimited)
<code>store.quotaexceededmsg</code>	Quota warning message. If none, notification is not sent. Default: None.
<code>store.quotaexceededmsginterval</code>	Interval, in days, for sending overquota notification. Default: 7
<code>store.quotagraceperiod</code>	Time, in hours, a mailbox has been overquota before messages to the mailbox will bounce back to the sender. Number of hours. Default: 120
<code>store.quotawarn</code>	Quota warning threshold. Percentage of quota exceeded before clients are sent an over quota warning. Default: 90
<code>local.store.quotaoverdraft</code>	Used to provide compatibility with systems that migrated from the Netscape Messaging Server. When ON, allow delivery of one message that puts disk usage over quota. After the user is over quota, messages are deferred or bounced, the quota warning message is sent, and the quota grace period timer starts. (The default is that the quota warning messages are sent when the message store reaches the threshold.) Default: Off, but is treated as on if <code>local.store.overquotastatus</code> is set, otherwise the user can never go over quota and the <code>overquotastatus</code> is never used.
<code>local.store.overquotastatus</code>	Enable quota enforcement before messages are enqueued in the MTA. This prevents the MTA queues from filling up. When set, and a user is not yet over quota, but an incoming message pushes the user over quota, then the message is delivered, but the <code>mailuserstatus</code> LDAP attribute is set to <code>overquota</code> so no more messages will be accepted by the MTA. Default: off

Message store quota also includes a couple of utilities. “[iminitquota](#)” in [Sun Java System Messaging Server 6.3 Administration Reference](#) initializes quota settings. In other words, quota

attributes and configutil parameters will take effect automatically after running this command. The changes would take effect without running this, but not immediately, as information is stored in caches and it will take a little time before the changes take effect.

The [imquotacheck](#) utility allows you to check message store usage against assigned quotas.

20.8.4 Configuring Message Store Quotas

This section describes the following tasks:

- “[20.8.4.1 To Specify a Default User Quota](#)” on page 620
- “[20.8.4.2 To Specify Individual User Quotas](#)” on page 621
- “[20.8.4.3 To Specify Domain Quotas](#)” on page 621
- “[To Setup Quota Notification](#)” on page 621
- “[20.8.4.4 To Enable or Disable Quota Enforcement](#)” on page 623
- “[20.8.4.5 To Set a Grace Period](#)” on page 624
- “[20.8.4.6 Netscape Messaging Server Quota Compatibility Mode](#)” on page 624

20.8.4.1 To Specify a Default User Quota

A default quota applies to users who do not have individual quotas set in their LDAP entries. The process consists of two steps: 1) Specifying a user default quota and 2) Specifying which users are bound to the default quota. The following examples show how to set default user quotas. Refer to Chapter 3, “[Messaging Server Configuration](#),” in *Sun Java System Messaging Server 6.3 Administration Reference* for detailed parameter information.

To specify a default user disk quota for message size in bytes:

```
configutil -o store.defaultmailboxquota -v [ -1 | number ]
```

where -1 indicates no quota (unlimited message usage) and *number* indicates a number of bytes.

To specify a default user quota for total number of messages:

```
configutil -o store.defaultmessagequota -v [ -1 | number ]
```

where -1 indicates no quota (unlimited messages) and *number* indicates the number of messages.

To specify the default quota for specific users:

Set the `mailQuota` attribute to -2 in the user entries that use the default message store quota. Note that if `mailQuota` is not specified, the system default quota is used.

20.8.4.2

To Specify Individual User Quotas

Each user can have individualized quotas. To set user-specific quotas, set the “`mailQuota`” in *Sun Java Communications Suite 5 Schema Reference* or “`mailMsgQuota`” in *Sun Java Communications Suite 5 Schema Reference* attributes in the user’s LDAP entry (see “`configutil Parameters`” in *Sun Java System Messaging Server 6.3 Administration Reference* for complete details). The following examples show how to set user quotas.

To specify the system default quota, do not add `mailQuota` to the LDAP entry, or set it to `-2`.

To set the quota to 1,000 messages set `mailMsgQuota` to `1000`.

To set the quota to two megabytes set `mailQuota` to `2M` or `2000000`.

To set the quota to two gigabytes, set `mailQuota` to `2G` or `2000000000` or `2000M`.

To specify a 2 Gigabyte quota; a 20 Megabyte voice mail quota; and a 100 Megabyte quota for the Archive folder:

```
mailQuota: 2G;#voice%20M;Archive%100M
```

The two gigabyte quota represents all folders in the user’s mailbox that are not explicitly assigned quotas. In this example, that excludes messages in the Archive folder, and messages of type voice. The 100 Megabyte quota includes messages in any folders within the Archive folder

20.8.4.3

To Specify Domain Quotas

You can set disk space or message quotas for domains. These quotas are for the cumulative bytes or messages of all users in a particular domain. To set domain quotas, set the “`mailDomainDiskQuota`” in *Sun Java Communications Suite 5 Schema Reference* or “`mailDomainMsgQuota`” in *Sun Java Communications Suite 5 Schema Reference* attributes in the desired LDAP domain entry..

To set the quota to 1,000 messages set `mailDomainMsgQuota` to `1000`.

To set the quota to two megabytes set `mailDomainDiskQuota` to `2M` or `2000000`.

To set the quota to two gigabytes, set `mailDomainDiskQuota` to `2G` or `2000000000` or `2000M`.

▼ To Setup Quota Notification

Quota notification is the process of sending users a warning message when they are getting close to their quota. Using this feature requires three steps.

1 Enable Quota Notification

Run the following at the command line:

```
configutil -o store.quotanotification -v [ yes | no ]
```

If the message is not set, no quota warning message is sent to the user.

2 Define a Quota Warning Message

The Warning Message is the message that will be sent to users who are close to exceeding their disk quota. To define a quota warning message at the command line:

```
configutil -o store.quotaexceededmsg -v 'message'
```

The message must be in RFC 822 format. It must contain a header with at least a subject line, followed by \$\$, then the message body. '\$' represents a new line. Depending on the shell that you are using, it might be necessary to append a \ before \$ to escape the special meaning of \$. (\$ is often the escape character for the shell.) Example:

```
configutil -o store.quotaexceededmsg -v "Subject: WARNING: User quota exceeded$$User quota threshold exceeded - reduce space used."
```

In addition, there is support for the following variables:

[ID] - userid

[DISKUSAGE] - disk usage

[NUMMSG] - number of messages

[PERCENT] - store.quotawarn percentage

[QUOTA] - mailquota attribute

[MSGQUOTA] - mailmsgquota attribute

Here's an example, using these variables:

```
configutil -o store.quotaexceededmsg -v "Subject: Overquota
Warning$$[ID],$$Your mailbox size has exceeded [PERCENT] of its allotted
quota.$Disk Usage: [DISKUSAGE]$Number of Messages: [NUMMSG]$Mailquota:
[QUOTA]$Message Quota: [MSGQUOTA]$$-Postmaster"
```

3 Specify how often the warning message is sent.

Set the following parameter:

```
configutil -o store.quotaexceededmsginterval -v number
```

where *number* indicates a number of days. For example, 3 would mean the message is sent every 3 days.

4 Specify a Quota Threshold

A quota threshold is a percentage of a quota that is exceeded before clients are sent a warning. When a user's disk usage exceeds the specified threshold, the server sends a warning message to the user.

Note – When `local.store.quotaoverdraft=on` email notifications are not triggered until the user's disk usage exceeds 100% of the quota regardless of the threshold set with `store.quotawarn`.

For IMAP users whose clients support the IMAP ALERT mechanism, the message is displayed on the user's screen each time the user selects a mailbox and a message is also written to the IMAP log.

To specify a quota threshold at the command line:

```
configutil -o store.quotawarn -v number
```

where *number* indicates a percentage of the allowed quota.

20.8.4.4

To Enable or Disable Quota Enforcement

By default, users or domains can exceed their quotas with no effect except for receiving an over quota notification (if set). Quota enforcement locks the mailboxes from receiving further messages until the disk usage is reduced below the quota level.

To enable or disable quota enforcement:

```
configutil -o store.quotaenforcement -v [ on | off ]
```

Note that over quota messages are saved in the MTA queues and a notification is sent to the sender stating that their messages was not delivered, but that a redelivery attempt will be made later. Delivery retries will continue until the grace period expires and all messages are sent back to the senders, or the disk usage falls below the quota and messages can be dequeued from the MTA and delivered to the message store. If you want to return messages that are over quota before they get to the message queues, use the following command line:

```
configutil -o local.store.overquotastatus -v on
```

To Enable Quota Enforcement at the Domain Level

To enforce quotas for a particular domain, use the command:

```
imquotacheck -f -d domain
```

To enable for all domains exclude the `-d` option. When a domain exceeds its quota, the `maildomainstatus` attribute is set to `overquota`, which halts all delivery to this domain. If a domain is not `overquota`, the value is set to `active`.

Disabling Quota Enforcement

If it appears that user quotas are being enforced, even when you have disabled them, check the following parameters:

These configutil parameters should be off or not set:

- store.quotaenforcement
- local.store.overquotastatus
- local.store.quotaoverdraft

Note that when local.store.overquotastatus is on, it always treats store.quotaoverdraft as on, otherwise the user will never go over quota to trigger the rejection. Also, when store.quotaoverdraft is on, the user is allowed one message which is smaller than the quota only. That is, it will never accept a message that is greater than the user's quota.

After making changes to these parameters, be sure to restart your messaging services.

These Message Store attributes should be active:

- maildomainstatus
- mailuserstatus

Note that messages will bounce if they are larger than the mailbox quota, regardless of quota enforcement configuration.

20.8.4.5

To Set a Grace Period

The grace period specifies how long the mailbox can be over the quota (disk space or number of messages) before messages are bounced back to sender. The grace period is not how long the message is held in the message queue, it's how long the mailbox is over quota before all incoming messages, including those in the message queue, are bounced. (see “[20.1 Overview](#)” on page 587 for more details.) The grace period starts when the user has reached the quota threshold and been warned. See “[To Setup Quota Notification](#)” on page 621.

To specify a quota grace period at the command line:

```
configutil -o store.quotagraceperiod -v number
```

where *number* indicates number of hours.

20.8.4.6

Netscape Messaging Server Quota Compatibility Mode

After disk usage exceeded the quota in the Netscape Messaging Server, the server deferred or bounced message delivery, sent an over quota notification, and started the grace period. Messaging Server provides a parameter, local.store.quotaoverdraft, which retains this behavior.

When set to ON, messages are delivered until disk usage is over quota. At that time, messages are deferred (messages stay in the MTA message queue but are not delivered to the message store), an over quota warning message is sent to the user, and a grace period starts. The grace period determines how long a mailbox is overquota before the overquota messages bounce. (The default is that the quota warning messages are sent when the message store reaches the threshold.) The default for this parameter is Off.

20.9 To Set the Automatic Message Removal (Expire and Purge) Feature

The automatic message removal feature (also known as expire and purge) automatically removes messages from the message store based on a set of administrator-defined criteria. This feature can automatically remove old or overly large messages, seen/deleted messages, messages with certain Subject: lines and so on. This feature allows the following removal criteria:

- By folder (mailboxes), users, domains, the entire message store, or specific partitions
- Number of messages in the mailbox
- Total size of the mailbox
- Age, in days, that a message has been in the mailbox
- Size of message and grace period (days that an oversized message will remain in the message store before purging)
- Whether a message has been flagged as *seen* or *deleted*
- Header strings

This feature is performed by the `imexpire` utility, which expunges and purges messages. See “[20.3 How the Message Store Removes Messages](#)” on page 593 for details on the message removal process.

Note – The server removes messages without warning, so it is important to inform users about automatic message removal policies. Unexpected message removal can be a source of consternation for users and administrators.

- “[20.9.1 imexpire Theory of Operation](#)” on page 625
- “[20.9.2 To Deploy the Automatic Message Removal Feature](#)” on page 626

20.9.1 imexpire Theory of Operation

`imexpire` can be invoked from the command line or scheduled to run automatically by the `imsched` daemon. The administrator specifies a set of expiration rules in a file called `store.expirerule`. This file specifies the criteria by which messages are removed. There can be multiple files with each put in the directory that pertains to the scope of the rules. That is, rules that apply globally to the entire message store are put in one directory, rules that apply to a partition in another, rules that apply to users in yet another, and so on.

Note – Although global expiration rules can be specified with the `configutil` command and `store.expire.attribute` parameters, it is better to use `store.expirerule` to specify these rules. If too many rules are created using `configutil`, performance problems can result.

`imexpire` loads all of the expire rules at start up. By default, `imexpire` creates one thread per partition. Each thread goes through the list of user folders under its assigned partition and loads the local expire rule files as it goes. The expire function checks each folder against the expire rules applicable to this folder and expunges messages as needed. If there is a `store.exp` file that exists under the mailbox directory, and there are messages that have been expunged/expired for longer than the time specified by the `store.cleanupage` configuration parameter, the purge function will permanently remove the message files under the message hash directories and remove the UID records from the `store.exp` files.

It is also possible to exclude specified users from the expire rules by adding their user ID, one per line, in a file called `expire_exclude_list` in `msg-svr-base/config/`.

20.9.2 To Deploy the Automatic Message Removal Feature

Automatic message removal requires three steps:

1. Define automatic message removal policy: Which messages will be automatically removed? What users, folders, domains and partitions will have messages automatically removed? What size, message age, headers will define the removal criteria. Define the scope of messages to be removed. See “[20.9.2.1 To Define Automatic Message Removal Policy](#)” on [page 626](#)
2. Specify the `imexpire` rules to implement this policy. See “[20.9.2.2 To Set Rules Implementing Automatic Message Removal Policy](#)” on [page 627](#)
3. Specify the `imexpire` scheduling. See “[20.9.2.3 To Schedule Automatic Message Removal and Logging Level](#)” on [page 633](#)

20.9.2.1 To Define Automatic Message Removal Policy

Define your automatic message removal policy by specifying the criteria for removal. `imexpire` allows for removal using the following criteria:

Age of Message. Automatically remove messages older than X days. Attribute: `messagedays`.

Message Count. Automatically remove messages in a folder exceeding X messages. Attribute: `messagecount`.

Age of Oversized Message. Automatically remove messages that exceed X bytes after Y days grace period. Attributes: `messagesize` and `messagesizedays`.

Seen and Deleted Message Flag. Automatically remove messages with the *Seen* or *Deleted* flag set. These criteria can be set to “and” or “or.” If set to *or*, the message’s *Seen/Delete* flag will cause automatic deletion regardless of other criteria. If set to *and*, the message’s *Seen/Delete* flag must be set along with passing all other specified criteria. Attributes: *seen* and *deleted*.

Header Field of Message. Allows you specify a header and string as criteria for removing a message. For example, removing all messages with the header “Subject: Work from Home!”

Folder of Messages. Allows you to specify the folder on which to remove messages. Attribute: *folderpattern*. Note that this attribute only uses the modified UTF-7 character set.

Note – *imexpire* does not allow you to delete or preserve messages based on how long it has been since that message was read. For example, you cannot specify that messages that have not been read for 200 days will be removed.

Examples of Automatic Message Removal Policy

Example 1: Remove all messages 365 days old in a folder exceeding 1,000 messages.

Example 2: Remove messages in domain siroe.com that are older than 180 days.

Example 3: Remove all messages that have been marked as *deleted*.

Example 4: Remove messages in sesta.com that have been marked as *seen*, are older than 30 days, are larger than 100 kilobytes, from folders exceeding 1,000 messages, with the header X-spam.

20.9.2.2

To Set Rules Implementing Automatic Message Removal Policy

To implement the automatic message removal policy defined in the previous section, you must set the *imexpire* rules. Rules are set by putting them into a *store.expirerule* file. An example of two global *store.expirerule* rules is shown below:

```
Rule1-regexp: 1
Rule1-folderpattern: user/.*/trash
Rule1-messagedays: 2
Rule2-regexp: 1
Rule2-folderpattern: user./*
Rule2-messagedays: 14
```

In this example, Rule 1 specifies that all messages in the trash folder will be remove after two days. Rule 2 specifies that all messages in the message store will be removed after 14 days.

This section consists of the following subsections:

- “Expiration Rules Guidelines” on page 628
- “Setting imexpire Rules Textually” on page 632
- “Setting imexpire Folder Patterns” on page 633

Expiration Rules Guidelines

This section sets the guidelines for the `store.expirerule` file rules.

Note – In earlier Messaging Server releases, expiration rules could be set with configutil parameters `store.expirerule.attribute` (see “[configutil Parameters](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.) This is still true, but expire rules using header constraints (example: expiring a message with a specific subject line) are not supported. Also, regular expressions in the expire rules created with configutil need to be POSIX compliant rules. If you want to use UNIX compliant regular expressions you will have to use the `store.expire` file. In addition, using both configutil options and the global `store.expirerule` configuration file is not supported. If the configuration file is present, configutil options are not used. In any case, it is best to use `store.expirerule` to specify all expiration rules.

- Rules are specified in a file called `store.expirerule`.
- Multiple expiration criteria can be specified with the same rule. (Example above.)
- Rules can apply to the entire message store (global rules), a partition, a user, or a folder.
 - The global rules are stored in `msg-svr-base/config/store.expirerule`

Note – Each global rule will be checked against every mailbox, which can cause some processing overhead depending on the number of global rules you specify. For this reason, you should not specify partition, mailbox or user rules in the global rules file. In general, you should try not to put any more expiration rules than necessary in this file.

- Partition rules are stored in `store_root/partition/partition_name/store.expirerule`.
- User rules are specified in `store_root/partition/partition_name/userid/store.expirerule` or by specifying the `folderpattern` rule to be `user/userid/*`
- Folder rules are specified in `store_root/partition/partition_name/userid/folder/store.expirerule` or by specifying the `folderpattern` rule to be `user/userid/folder`
- Note that multiple non-global rules (user, folder, partition) using `rule_name` was only implemented in the Messaging Server 6.2p4 release and later.
- Multiple expire rules can be applied to a mailbox at the same time. An expire policy for a mailbox consists of global rules and local rules. Local rules apply to the mailbox under the same directory and all of its sub-folders.

- `imexpire` unifies all of the expiration rules applying to a mailbox, unless there is an exclusive rule specified for this mailbox (see [Table 20–8](#)). The resulting rule set represents the most restrictive expiration policy based on all applicable rules. For example, if rule X expires messages such that the maximum message life is 10 days, and rule Y specifies 5 days, the union will be 5 days.

TABLE 20–8 `imexpire` Attributes

Attribute	Description (Attribute Value)
<code>action</code>	Specifies an action to perform on the messages caught by the expire rules. The possible values are: <code>discard</code> discards the message. This is the default. <code>report</code> action prints the mailbox name, uid-validity and uid to stdout. <code>archive</code> archives the message with Sun Compliance and Content Management System and then discards the message. <code>fileinto: folder</code> action folders the message into the specific folder. The shared folder prefix can be used to folder messages to folders owned by another user.
<code>exclusive</code>	Specifies whether or not this is an exclusive rule. If specified as <code>exclusive</code> , only this rule applies to the specified mailbox(s) and all other rules are ignored. If more than one exclusive rule exists, the last exclusive rule loaded will be used. For example, if a global and a local exclusive rule are specified, the local rule will be used. If there is more than one global exclusive rule, the last global rule listed by <code>configutil</code> is used. (1/0)
<code>folderpattern</code>	Specifies the folders affected by this rule. The format must start with a <code>user/</code> , which represents the directory <code>store_root/partition/*</code> See Table 20–9 . (POSIX regular expression)
<code>messagecount</code>	Maximum number of messages in a folder. Oldest messages are expunged as additional messages are delivered. (integer)
<code>foldersize</code>	Maximum size of folder before the oldest messages are expunged when additional messages are delivered. (integer in bytes)
<code>messagedays</code>	Number of days in the message store before being expunged. (integer)
<code>messagesize</code>	Maximum size of message in bytes before it is marked to be expunged. (integer)
<code>messagesizedays</code>	Grace period. Days an over-sized message should remain in a folder. (integer)
<code>messageheader.header</code>	Specifies a header field and string that marks a message for removal. Values are not case-sensitive and regular expressions are not recognized. Example: Rule1.messageheader.Subject: Get Rich Now! For the header <code>Expires</code> and <code>Expiry-Date</code> , <code>imexpire</code> will remove the message if the date value specified with these header fields is older than the <code>messagedays</code> attribute. If multiple expiration header fields are specified, the earliest expiration date will be used. (string).
<code>regexp</code>	Enable UNIX regular expressions in rules creation. (1 or 0). If not specified, IMAP expressions will be used.
<code>savedays</code>	Number of days the messages are saved in a folder until they are expunged.

TABLE 20-8 imexpire Attributes (*Continued*)

Attribute	Description (Attribute Value)
seen	seen is a message status flag set by the system when the user opens a message. If the attribute seen is set to and, then the message must be seen and other criteria must be met before the rule is fulfilled. If the attribute seen is set to or, then the message only needs to be seen or another criteria be met before the rule is fulfilled. (and/or).
sieve	A Sieve rule specifying message selection criteria. Example: Rule17.sieve: header :contains "Subject" "Vigara"
deleted	deleted is a message status flag set by the system when the user deletes a message. If the attribute deleted is set to and, then the message must be deleted and another criteria must be met before the rule is fulfilled. If the attribute deleted is set to or, then the message only needs to be seen or another criteria be met before the rule is fulfilled. (and/or)

Localized Mailbox Names

The IMAP protocol specifies that mailbox names use modified UTF-7 encoding. Messaging Server supports localized character sets on external interfaces so that mailbox names can be localized. Internally, however, the system converts the localized name to UTF-7. Thus, a folder that has a localize mailbox name on a client will have a corresponding mailbox file name in UTF-7. (Note that IMAP error messages will output mailbox names in UTF-7 and not the localized character set.)

In general, most message store utilities that require mailbox names expect the names in the localized character set, although they may have an option flag that allows a different character set to be used. These utilities include `reconstruct`, `mboxutil`, `imsbackup`, `imsrestore`, and `hashdir`. However, `imexpire` requires that the mailbox name, specified as the attribute `folderpattern`, be in UTF-7. Using a localized name will not work.

To obtain the appropriate `folderpattern` for `imexpire` it may be necessary to convert a localized mailbox name to the modified UTF-7 equivalent. This can be done using the `mboxutil -E` command as follows:

```
$ mboxutil -l -o user/user1/*
msgs Kbytes last msg      partition quotaroot mailbox
|
77    27   2006/9/9 3:21 primary  10240 user/kat/INBOX
0      0   -           - primary      - user/kat/箱

$ mboxutil -l -E M-UTF7 -P user/user1/*
msgs Kbytes last msg      partition quotarcot mailbox

77    27   2006/9/9 3:21 primary  10240 user/kat/INBOX
0      0   -           - primary      - user/kat'&V4NXPnux-
```

The first `mboxutil` shows the localized filename. The second `mboxutil` shows the filename in modified UTF-7. It is also possible to use the IMAP list command:

```
2 list "" *
* LIST (\NoInferiors) "/" INBOX
* LIST (\HasNoChildren) "/" &V4NXPnux-
```

To convert the local charset to modified UTF-7 encoding, use the `mboxutil` command with the `-E` option:

```
$ mboxutil -l -E M-UTF7 -P user/han/送信
msgs  Kbytes last msg      partition quotar
                                |
0      0   -           -      lab
```

Note that `mboxutil -E` can be used for any command that requires the use of a UTF-7 mailbox name including `imexplore`.

Setting imexpire Rules Textually

Automatic message removal rules are set by specifying rules in a `store.expirerule` file. The `store.expirerule` file contains one expire criteria per line. An expire criteria of the global rule configuration file (`msg-svr-base/data/store/store.expirerule`) has the following format:

rule_name.attribute: value

An expiration rule for a user or mailbox rule configuration file has this format:

attribute: value

[Example 20–4](#) shows a set of global expiration rules in `msg-svr-base/config/store.expirerule`.

Rule 1 sets the global expiration policy (that is, policy that applies to all messages), as follows:

- Enable UNIX regular expressions in rules creation.
- Removes messages larger than 100,000 bytes after 3 days.
- Removes messages deleted by the user.
- Removes any message with the strings “Vigara Now!” or “XXX Porn!” in the Subject: header.
- Limits all folders to 1,000 messages. After 1,000 messages, the system removes the oldest messages on a folder to keep the total to 1,000.
- Removes all messages older than 365 days.

Rule 2 sets the automatic message removal policy for users at the hosted domain `siroe.com`. It limits mailbox sizes to 1 megabyte, removes messages that have been deleted, and removes messages older than 14 days.

Rule 3 sets the automatic message removal policy for messages in the `inbox` folder of user `f.dostoevski`. It removes messages with a subject line having the expression “On-line Casino.”

EXAMPLE 20–4 Example imexpire Rules

```
Rule1-regexp: 1
Rule1-folderpattern: user/.*
Rule1-messagesize: 100000
Rule1-messagesizedays: 3
Rule1-deleted: or
Rule1-Subject: Vigara Now!
Rule1-Subject: XXX Porn!
Rule1-messagecount: 1000
Rule1-messagedays: 365
Rule2-regexp: 1
Rule2-folderpattern: user/.*@siroe.com/.*
Rule2-exclusive: 1
```

EXAMPLE 20-4 Example imexpire Rules *(Continued)*

```
Rule2.deleted: or
Rule2.messagedays: 14
Rule2.messagecount: 1000
Rule3.folderpattern: user/f.dostoevski/inbox
Rule3.Subject: *On-line Casino*
```

Setting imexpire Folder Patterns

Folder patterns can be specified using POSIX regular expressions by setting the `imexpire` attribute `regex` to 1. If not specified, IMAP expressions will be used. The format must start with a `user/` followed by a pattern. [Table 20-9](#) shows the folder pattern for various folders.)

TABLE 20-9 imexpire Folder Patterns Using Regular Expressions

Folder Pattern	Scope
<code>user/userid/.*</code>	Apply rule to all messages in all folders of <code>userid</code> .
<code>user/userid/Sent</code>	Apply rule to messages of <code>userid</code> in folder Sent:
<code>user/.*</code>	Apply rule to entire message store.
<code>user/.*/trash</code>	Apply rule to the trash folder of all users.
<code>user/.*@siroe.com/.*</code>	Apply rule to folders in hosted domain siroe.com:
<code>user/[^@]*/*</code>	Apply rule to folders in default domain.

20.9.2.3**To Schedule Automatic Message Removal and Logging Level**

Automatic message removal is activated by the `imsched` scheduling daemon. By default, `imsched` invokes `imexpire` at 23:00 every day and messages are both expunged and purged. This schedule can be customized by setting the `configutil` parameter `local.schedule.expire` and `store.cleanupage` described in [Table 20-10](#).

Expire and purge can take a long time to complete on a large message store, so you may wish to experiment and decide how often to run these processes. For example, if an expire/purge cycle takes 10 hours, you may not want the default schedule of running expire and purge once a day. Schedule expire and purge using the `imexpire` command and the automatic task scheduling parameter (see “[4.6 To Schedule Automatic Tasks](#)” on page 129). For example:

```
configutil -o local.schedule.expire -v "0 1 * * 6 /opt/SUNWmsgsr/sbin/imexpire -e"
configutil -o local.schedule.mspurge -v "0 23 * * * /opt/SUNWmsgsr/sbin/imexpire -c"
```

In this example, messages are expired at 1AM Saturdays and purged every night at 11PM. If no purge schedule is set, `imexpire` will perform purge after an expire.

TABLE 20–10 Expire and Purge configutil Log and Scheduling Parameters

Parameter	Description
<code>local.schedule.expire</code>	<p>Interval for running <code>imexpire</code>. Uses UNIX crontab format: <i>minute hour day-of-month month-of-year day-of-week</i></p> <p>The values are separated by a space or tab and can be 0-59, 0-23, 1-31, 1-12 and 0-6 (with 0=Sunday) respectively. Each time field can be either an asterisk (meaning all legal values), a list of comma-separated values, or a range of two values separated by a hyphen. Note that days can be specified by both day of the month and day of the week, however, it is not typical to use them both since the number of such occurrences are very small. If they are both specified, then both will be required. For example, setting the 17th day of the month and Tuesday will require both values to be true.</p> <p>Note that you can also use the -e and -c flags with <code>imexpire</code> to and expire only or purge only respectively. See “imexpire” in <i>Sun Java System Messaging Server 6.3 Administration Reference</i>.</p> <p>Interval Examples:</p> <ol style="list-style-type: none"> 1) Run <code>imexpire</code> at 12:30am, 8:30am, and 4:30pm: <code>30 0,8,16 * * *</code> /opt/SUNWmsgsr/sbin/imexpire 2) Run <code>imexpire</code> at weekday morning at 3:15 am: <code>15 3 * * 1-5</code> /opt/SUNWmsgsr/sbin/imexpire 3) Run <code>imexpire</code> only on Mondays: <code>0 0 * * 1</code> /opt/SUNWmsgsr/sbin/imexpire <p>Default:</p> <pre>0 23 * * * /opt/SUNWmsgsr/sbin/imexpire</pre> <p>To disable: set <code>local.schedule.expire.enable</code> to NO.</p>
<code>store.cleanupage</code>	<p>Age (in hours) of expired or expunged message before purge will permanently remove it.</p> <p>Default: None</p>
<code>local.store.expire.loglevel</code>	<p>Specify a log level:</p> <p>1 = log summary for the entire expire session. 2 = log one message per mailbox expired. 3 = log one message per message expired.</p> <p>Default: 1</p>

Setting imexpire Logging Levels

`imexpire` will log a summary to the default log file upon completion. If `expire` is invoked from the command line, the `-v` (verbose) and `-d` (debug) option can be used to instruct `imexpire` to log detail status/debug messages to stderr. If `imexpire` is invoked by `imsched`, the configutil parameter `local.store.expire.loglevel` can be set to 1,2 or 3 for different levels of logging. Loglevel 1 is the default, it will log a summary for the entire expire session. Loglevel 2 will log one message per mailbox expired. Loglevel 3 will log one message per message expired.

Excluding Specified Users from Automatic Message Remove

Exclude specified users from the expire rules by adding their user ID, one per line, in a file called `expire_exclude_list` in `msg-svr-base/config/`. Or, configure a dummy exclusive expire rule under the user's mailbox.

20.10 Configuring Message Store Partitions

Mailboxes are stored in message store partitions, an area on a disk partition specifically devoted to storing the message store. Message store partitions are not the same as disk partitions, though for ease of maintenance, it is recommended that you have one disk partition and one file system for each message store partition. Message store partitions are directories specifically designated as a message store.

User mailboxes are stored by default in the `store_root/partition/` directory (see “[20.2 Message Store Directory Layout](#)” on page 589). The `partition` directory is a logical directory that might contain a single partition or multiple partitions. At start-up time, the `partition` directory contains one subpartition called the primary partition.

You can add partitions to the `partition` directory as necessary. For example, you might want to partition a single disk to organize your users as follows:

```
store_root/partition/mktng/  
store_root/partition/eng/  
store_root/partition/sales/
```

As disk storage requirements increase, you might want to map these partitions to different physical disk drives.

You should limit the number of mailboxes on any one disk. Distributing mailboxes across disks improves message delivery time (although it does not necessarily change the SMTP accept rate). The number of mailboxes you allocate per disk depends on the disk capacity and the amount of disk space allocated to each user. For example, you can allocate more mailboxes per disk if you allocate less disk space per user.

If your message store requires multiple disks, you can use RAID (Redundant Array of Inexpensive Disks) technology to ease management of multiple disks. With RAID technology,

you can spread data across a series of disks but the disks appear as one logical volume so disk management is simplified. You might also want to use RAID technology for redundancy purposes; that is, to duplicate the store for failure recovery purposes.

Note – To improve disk access, the message store and the message queue should reside on separate disks.

20.10.1

To Add a Partition

When adding a partition, you specify both an absolute physical path where the partition is stored on disk and a logical name, called the partition nickname.

The partition nickname allows you to map users to a logical partition name regardless of the physical path. When setting up user accounts and specifying the message store for a user, you can use the partition nickname. The name you enter must be an alphanumeric name and must use lowercase letters.

To create and manage the partition, the user ID used to run the server must have permission to write to the location specified in the physical path.

Note – After adding a partition, you must stop then restart the server to refresh the configuration information.

▼ To Add a Message Store Partition

- **Command Line, To add a partition to the store at the command line:**

```
configutil -o store.partition.nickname.path -v path
```

where *nickname* is the logical name of the partition and *path* indicates the absolute path name where the partition is stored.

To specify the path of the default primary partition:

```
configutil -o store.partition.primary.path -v path
```

20.10.2

To Move Mailboxes to a Different Disk Partition

By default, mailboxes are created in the primary partition. If the partition gets full, additional messages cannot be stored. There are several ways to address the problem:

- Reduce the size of user mailboxes
- If you are using volume management software, add additional disks
- Create additional partitions (“[20.10.1 To Add a Partition](#)” on page 636) and move mailboxes to the new partitions

If possible, we recommend adding additional disk space to a system using volume management software since this procedure is the most transparent for the user. However, you may also move mailboxes to a different partition.

▼ To Move Mailboxes to a Different Disk Partition

- 1 **The user does not have to be disconnected from their mailbox during this migration process.**
mboxutil -r should lock everything that needs to be locked. Delivery may be stalled while that happens, and POP and IMAP and therefore webmail clients, may experience a delay, but there should be no problem.
- 2 **Move the user mailbox with the following command:**
`mboxutil -r user/<userid>/INBOX user/<userid>/INBOX <partition_name>`
Example:
`mboxutil -r user/ofanning/INBOX user/ofanning/INBOX secondary`
- 3 **Set the `mailMessageStore` attribute in the moved user's LDAP entry to the name of the new partition.**
Example: `mailMessageStore: secondary`
- 4 **Inform the user that message store connection is now allowed. If applicable, change the `mailAllowedServiceAccess` attribute to allow POP, IMAP and HTTP services.**

20.10.3 Changing the Default Message Store Partition Definition

The default partition is the partition used when a user is created and the `mailMessageStore` LDAP attribute is not specified in the user entry. The `mailMessageStore` LDAP attribute, which specifies a user's message store partition, should be specified in all user entries so that a default partition is not necessary. In addition, the default partition should **not** be changed for load balancing or any other reason. It is invalid and dangerous to change the default partition while there are still users depending on the default partition definition.

If it is absolutely necessary to change the default partition, make sure that all users on the old default partition (the one being left behind) have their `mailMessageStore` attribute set to their current partition (which will no longer be the default), before changing the definition of default with the `configutil` parameter `store.defaultpartition`.

20.11 Performing Message Store Maintenance Procedures

This section provides information about the utilities you use to perform maintenance and recovery tasks for the message store. You should always read your postmaster mail for warnings and alerts that the server might send. You should also monitor the log files for information about how the server is performing. For more information about log files, see [Chapter 25, “Managing Logging”](#)

This section contains the following:

- “[20.11.1 Adding More Physical Disks to the Message Store](#)” on page 638
- “[20.11.2 To Manage Mailboxes](#)” on page 638
- “[20.11.3 Maximum Mailbox Size](#)” on page 641
- “[20.11.4 To Monitor Quota Limits](#)” on page 641
- “[20.11.5 To Monitor Disk Space](#)” on page 642
- “[20.11.6 The stored Daemon](#)” on page 642
- “[20.11.7 Reducing Message Store Size Due to Duplicate Storage of Identical Messages](#)” on page 643

20.11.1 Adding More Physical Disks to the Message Store

The Messaging Server message store contains the user mailboxes for a particular Messaging Server instance. The size of the message store increases as the number of mailboxes, folders, and log files increase.

As you add more users to your system, your disk storage requirements increase. Depending on the number of users your server supports, the message store might require one physical disk or multiple physical disks. Messaging Server enables you to add more stores as needed. One approach to adding more stores is by using storage appliances. See [Using NetApp Filers with Sun Java System Messaging Server Message Store](#) for information on how to configure Network Appliance storage appliances with Messaging Server.

20.11.2 To Manage Mailboxes

This section describes the following utilities for managing and monitoring mailboxes: `mboxutil`, `hashdir`, `readership`.

20.11.2.1 The `mboxutil` Utility

Use the `mboxutil` command to perform typical maintenance tasks on mailboxes. `mboxutil` tasks include the following:

- List mailboxes
- List and remove orphaned and inactive mailboxes

- Create mailboxes
- Rename mailboxes
- Move mailboxes from one partition to another
- Expunge mailboxes
- Restore expunged messages that have not been purged
- List personal mailbox subscriptions and unsubscribed mailboxes that no longer exist
- You can also use the `mboxutil` command to view information about quotas. For more information, see “[20.11.4 To Monitor Quota Limits](#)” on page 641.

Note – Note that you should not kill the `mboxutil` process in the middle of execution. If it is killed with `SIGKILL` (`kill -9`), it may potentially require that every server get restarted and a recovery be done.

See “[mboxutil](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for detailed syntax and usage requirements.

Examples

To list all mailboxes for all users:

```
mboxutil -l
```

To list all mailboxes and also include path and ACL information:

```
mboxutil -l -x
```

To create the default mailbox named `INBOX` for the user `daphne`:

```
mboxutil -c user/daphne/INBOX
```

To delete a mail folder named `projx` for the user `delilah`:

```
mboxutil -d user/delilah/projx
```

To delete the default mailbox named `INBOX` and *all mail folders* for the user `druscilla`:

```
mboxutil -d user/druscilla/INBOX
```

To rename the mail folder `memos` to `memos-april` for the user `desdemona`:

```
mboxutil -r user/desdemona/memos user/desdemona/memos-april
```

To move the mail account for the user `dimitria` to a new partition:

```
 mboxutil -r user/dimitria/INBOX user/dimitria/INBOX partition
```

where *partition* specifies the name of the new partition.

To move the mail folder named **personal** for the user **dimitria** to a new partition:

```
 mboxutil -r user/dimitria/personal user/dimitria/personal partition
```

20.11.2.2

To Remove Orphan Accounts

To search for orphaned accounts (orphaned accounts are mailboxes that do not have corresponding entries in LDAP) use the following command:

```
 mboxutil -o
```

Command output follows:

```
 mboxutil: Start checking for orphaned mailboxes
user/annie/INBOX
user/oliver/INBOX
mboxutil: Found 2 orphaned mailbox(es)
mboxutil: Done checking for orphaned mailboxes
```

Use the following command to create a file listing orphaned mailboxes that can be turned into a script file that deletes the orphaned mailboxes (example filename is **orphans.cmd**):

```
 mboxutil -o -w orphans.cmd
```

The command output is as follows:

```
 mboxutil: Start checking for orphaned mailboxes
mboxutil: Found 2 orphaned mailbox(es)
mboxutil: Done checking for orphaned mailboxes
```

Delete the orphan files with the following command:

```
 mboxutil -d -f orphans.cmd
```

20.11.2.3

The **hashdir** Utility

The mailboxes in the message store are stored in a hash structure for fast searching. Consequently, to find the directory that contains a particular user's mailbox, use the **hashdir** utility.

This utility identifies the directory that contains the message store for a particular account. This utility reports the relative path to the message store, such as **d1/a7/**. The path is relative to the directory level just before the one based on the user ID. The utility sends the path information to the standard output.

For example, to find the relative path to the mailbox for user crowe:

```
hashdir crowe
```

20.11.2.4 The readership Utility

The readership utility reports on how many users other than the mailbox owner have read messages in a shared IMAP folder.

An owner of a IMAP folder may grant permission for others to read mail in the folder. A folder that others are allowed to access is called a *shared folder*. Administrators can use the readership utility to see how many users other than the owner are accessing a shared folder.

This utility scans all mailboxes and produces one line of output per shared folder, reporting the number of readers followed by a space and the name of the mailbox.

Each reader is a distinct authentication identity that has selected the shared folder within the past specified number of days. Users are not counted as reading their own personal mailboxes. Personal mailboxes are not reported unless there is at least one reader other than the folder's owner.

For example, the following command counts as a reader any identity that has selected the shared IMAP folder within the last 15 days:

```
readership -d 15
```

20.11.3 Maximum Mailbox Size

A mailbox has a maximum size of about one million messages. More than this will cause messages to stop being delivered to the user and could cause message store performance problems. See “[20.14.4.8 User Mail Not Delivered Due to Mailbox Overflow](#)” on page 674 for details.

20.11.4 To Monitor Quota Limits

Monitor quota usage and limits by using `imquotacheck`, which generates a report listing defined quotas and limits, and provides information on quota usage. Quotas and usage figures are reported in kilobytes. This utility can also compare mailbox size with a user's assigned quota. As an option, you can email a notification to users who have exceeded a set percentage of their assigned quota.

Note – Certain functions have changed in `imquotacheck`. (In Messaging Server 6.x, `imquotacheck` utility has superseded the `quotacheck` utility.) In Messaging Server 5.x, when you used the `quotacheck` utility to retrieve a list of users, `quotacheck` searched the local `mboxlist` database. This function duplicated the search function in the `mboxutil` utility.

In Messaging Server 6.x, this duplicate function was removed from the `imquotacheck` utility. If you perform a user search with `imquotacheck`, the search is performed against the LDAP directory, not the local `mboxlist` database. To retrieve a list of users from the local `mboxlist` database, use the `mboxutil` utility.

To list the usage of all users whose quota exceeds the least threshold in the rule file:

```
imquotacheck
```

List quota information for a the domain `siroe.com`:

```
imquotacheck -d siroe.com
```

To send a notification to all users in accordance to the default rule file:

```
imquotacheck -n
```

To send a notification to all users in accordance to a specified *rulefile*, *myrulefile*, and to a specified mail template file, *mytemplate.file* (for more information, refer to “[imquotacheck](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*):

```
imquotacheck -n -r myrulefile -t mytemplate.file
```

To list per folder usages for one user `user1` (will ignore the rule file):

```
imquotacheck -u user1 -e
```

20.11.5 To Monitor Disk Space

You can specify how often the system should monitor disk space and partition usage, and under what circumstances the system should send a warning. See “[27.3.2 Monitoring Disk Space](#)” on page 869 for details.

20.11.6 The stored Daemon

The stored daemon performs the following maintenance tasks for the message store:

- Performs checkpoint database transactions..
- Deadlock detection and rollback of deadlocked database transactions.

- Cleanup of temporary files and lock files on startup.
- Creates a database snapshot archive
- Database recovery as necessary (see “[20.14.2 Message Store Startup and Recovery](#)” on page [664](#))

If any server daemon crashes, you must stop all daemons and restart all daemons including stored.

20.11.7 Reducing Message Store Size Due to Duplicate Storage of Identical Messages

When a message is sent to multiple recipients, that message is placed in each recipient’s mailbox. Some messaging systems store separate copies of the same message in each recipient’s mailbox. By contrast, the Sun Java System Messaging Server strives to retain a single copy of a message regardless of the number of mailboxes in which that message resides. It does this by creating hard links to that message in the mailboxes containing that message.

When other messaging systems are migrated to the Sun Java Messaging Server, these multiple message copies may be copied over with the migration process. With a large message store, this means that a lot of messages are duplicated unnecessarily. In addition, multiple copies of the same message can be accumulated in normal server operation, for example, from IMAP append operations or other sources.

Messaging Server provides a new command called relinker that removes the excess message copies and replaces them with hard links to a single copy.

20.11.7.1 Relinker Theory of Operations

The relinking function can be run in the command or realtime mode. When the relinker command is run, it scans through the message store partitions, creates or updates the MD5 message digest repository (as hard links), deletes excess message files, and creates the necessary hard links.

The digest repository consists of hard links to the messages in the message store. It is stored in the directory hierarchy *partition_path*=md5. This directory is parallel to the user mailbox hierarchy *partition_path*=user (see [Figure 20–1](#)). Messages in the digest repository are uniquely identified by their MD5 digest. For example, if the digest for fredb/00/1.msg is 4F92E5673E091B43415FFFA05D2E47, then *partition*=user/*hashdir*/*hashdir*=fredb/00/1.msg is linked to *partition*=md5/*hashdir*/*hashdir*/4F92E5673E091B43415FFFA05D2E47EA.msg. If another mailbox has this same message, for example, *partition_path*=user/*hashdir*/*hashdir*/gregk/00/17.msg, that message will also be hard linked to *partition_path*=md5/4F/92/4F92E5673E091B43415FFFA05D2E47EA.msg. This is shown in [Figure 20–4](#).

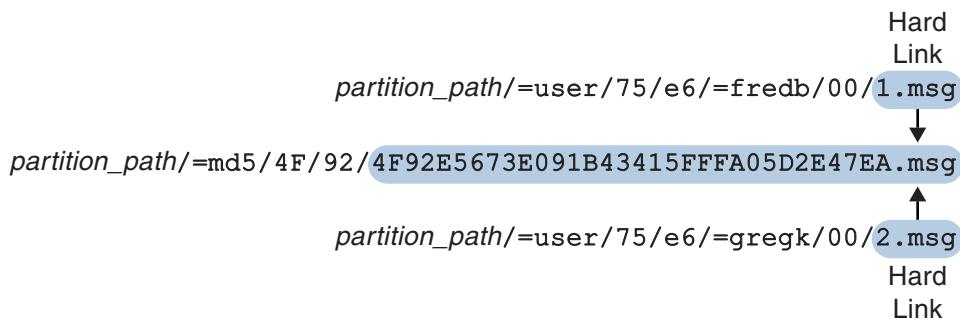


FIGURE 20–4 Message Store Digest Repository

For this message, the link count will be three. If both messages are deleted from the mailboxes of fredb and gregk, then the link count will be one and the message can be purged.

The relinker process can also be run in the realtime mode for similar functionality. See “[20.11.7.3 Using Relinker in the Realtime Mode](#)” on page [645](#) for details.

20.11.7.2 Using Relinker in the Command Line Mode

relinker scans through a message store partition, creates or updates the MD5 message repository (as hard links) and deletes excess message files. After relinker scans a store partition, it outputs statistics on the number of unique messages and size of the partition before and after relinking. To run more quickly on an already hashed store, relinker only computes the digest of the messages not yet present in =md5. It also has an option to erase the entire digest repository (which doesn’t affect the user mailboxes).

The syntax for the command is as follows:

`relinker [-P. partitionname] [-d]`

where *partitionname* specifies the partition to be processed (default: all partitions) and -d specifies that the digest repository be deleted. Sample output is shown below:

```

# relinker

Processing partition: primary
Scanning digest repository...
Processing user directories.....
-----
Partition statistics      Before      After
-----
Total messages           4531898    4531898
Unique messages          4327531    3847029
Message digests in repository   0        3847029
Space used               99210Mb   90481Mb

```

```

Space savings from single-copy      3911Mb      12640Mb
-----
# relinker -d
Processing partition: primary
Purging digest repository...
-----
Partition statistics           Before     After
-----
Message digests in repository    3847029      0
-----
```

Relinker can take a long time to run, especially for the first time if there are no messages are in the repository. This is because it has to calculate the digest for every message (if the relinker criteria is configured to include all messages—see “[20.11.7.4 Configuring Relinker](#)” on page 646 for information on configuring relinker criteria.) For example, it could take six hours to process a 100 Gigabyte message store. However, if run-time relinking is enabled see “[20.11.7.3 Using Relinker in the Realtime Mode](#)” on page 645.

If the `relinker` command line mode is used exclusively, and not the run-time option, it is necessary to purge the digest repository (=md5), otherwise messages purged in the store (=user) will not become available disk space since they still have a link in the digest repository (they become orphaned). If you are just performing a one-time optimization of the store—for example after a migration—you can run `relinker` once, then delete the entire repository with `relinker -d`. For repeated purging during migration, it is sufficient to just run the `relinker` command repeatedly, since each time it runs it also purges the expired or orphaned messages from the repository.

It is safe to run multiple instances of `relinker` in parallel with each processing a different partition (using the `-p` option). Messages are only relinked inside the same partition.

20.11.7.3

Using Relinker in the Realtime Mode

The `relinker` function can be enabled in the realtime mode by setting the `configutil` parameter `local.store.relinker.enabled` to `yes`. Using `relinker` in the realtime mode will compute the digest of every message delivered (or restored, IMAP appended, and so forth) which matches the configured relinker criteria (“[20.11.7.4 Configuring Relinker](#)” on page 646), then look in the repository to see if that digest is already present. If the digest is present, it creates a link to it in the destination mailbox instead of creating a new copy of the message. If there is no digest, it creates the message and adds a link to it in the repository afterwards.

`stored` scans the digest repositories of each partition and purges the messages having a link count of 1, or which don’t match the relinker criteria. The scan is done one directory at a time over a configurable time period. This is so that the I/O load is evenly distributed and doesn’t noticeably impact other server operations. By default the purge cycle is 24 hours, which means

messages can still be present on the disk for up to 24 hours after they've been deleted from the store or have exceeded the configured maximum age. This task is enabled when the relinker realtime mode is enabled.

20.11.7.4 Configuring Relinker

[Table 20–11](#) shows the parameters used to set relinker criteria.

TABLE 20–11 `relinker configutil` Parameters

Parameter	Description
<code>local.store.relinker.enabled</code>	Enables real-time relinking of messages in the append code and <code>stored</code> purge. The <code>relinker</code> command-line tool may be run even if this option is off. However since <code>stored</code> will not purge the repository, <code>relinker -d</code> must be used for this task. Turning this option on affects message delivery performance in exchange for the disk space savings. Default: no
<code>local.store.relinker.maxage</code>	Maximum age in hours for messages to be kept in the repository, or considered by the <code>relinker</code> command-line. -1 means no age limit, that is, only purge orphaned messages from the repository. For <code>relinker</code> it means process existing messages regardless of age. Shorter values keep the repository smaller thus allow <code>relinker</code> or <code>stored</code> purge to run faster and reclaim disk space faster, while longer values allow duplicate message relinking over a longer period of time, for example, when users copy the same message to the store several days apart, or when running a migration over several days or weeks. Default: 24
<code>local.store.relinker.minsize</code>	Minimum size in kilobytes for messages to be considered by run-time or command-line <code>relinker</code> . Setting a non-zero value gives up the relinker benefits for smaller messages in exchange for a smaller repository. Default: 0
<code>local.store.relinker.purgecycle</code>	Approximate duration in hours of an entire <code>stored</code> purge cycle. The actual duration depends on the time it takes to scan each directory in the repository. Smaller values will use more I/O and larger values will not reclaim disk space as fast. 0 means run purge continuously without any pause between directories. -1 means don't run purge in <code>stored</code> (then purge must be performed using the <code>relinker -d</code> command). Default: 24

20.12 Backing Up and Restoring the Message Store

Message store backup and restore is one of the most common and important administrative tasks. It consists of backing up all the messages and folders on a message store. You must implement a backup and restore policy for your message store to ensure that data is not lost if problems such as the following occur:

- System crashes

- Hardware failure
- Accidental deletion of messages or mailboxes
- Problems when reinstalling or upgrading a system
- Natural disasters (for example, earthquakes, fire, hurricanes)
- Migrating users

You can do message store back up and restore using command-line utilities `imsbackup` and `imsrestore`, or the integrated solution that uses Legato Networker™.

Messaging Server provides a single-copy backup procedure. Regardless of how many user folders contain a particular message, during backup, the message file is backed up only once using the first message file found. The second message copy is backed up as a link to the name of the first message file, and so on. `imsbackup` maintains a hash table of all messages using the device and inode of the message files as the index. This method does have implications when restoring data, however. For more information, see [“20.12.5 Considerations for Partial Restore” on page 651](#)

Note – It is also possible to do message store backup and restore by backing up all the message files and directories. Refer to [“20.12.9 Message Store Disaster Backup and Recovery” on page 658](#).

This section contains the following subsections:

- [“20.12.1 Creating a Mailbox Backup Policy” on page 647](#)
- [“20.12.2 To Create Backup Groups” on page 648](#)
- [“20.12.3 Messaging Server Backup and Restore Utilities” on page 649](#)
- [“20.12.4 Excluding Bulk Mail When You Perform Backups” on page 650](#)
- [“20.12.5 Considerations for Partial Restore” on page 651](#)
- [“20.12.6 To Use Legato Networker” on page 653](#)
- [“20.12.7 To Use a Third Party Backup Software \(Besides Legato\)” on page 655](#)
- [“20.12.8 Troubleshooting Backup and Restore Problems” on page 657](#)
- [“20.12.9 Message Store Disaster Backup and Recovery” on page 658](#)

20.12.1 Creating a Mailbox Backup Policy

Your backup policy will depend on several factors, such as:

- [“20.12.1.1 Peak Business Loads” on page 648](#)
- [“20.12.1.2 Full and Incremental Backups” on page 648](#)
- [“20.12.1.3 Parallel or Serial Backups” on page 648](#)

20.12.1.1 Peak Business Loads

You need to take into account peak business loads when scheduling backups for your system as this can reduce system load during peak hours. For example, backups are probably best scheduled for early morning hours such as 2:00 AM.

20.12.1.2 Full and Incremental Backups

Incremental backups (see “[Incremental Backup](#)” on page 650) will scan the store for changed data and back up only what has changed. Full backups will back up the entire message store. You need to determine how often the system should perform full as opposed to incremental backups. You’ll probably want to perform incremental backups as a daily maintenance procedure and full backups once a week.

20.12.1.3 Parallel or Serial Backups

When user data is stored on multiple disks, you can back up user groups in parallel if you wish. Depending on system resources, parallel backups can speed up the overall backup procedure. However, you might want to use serial backups if you want to reduce backup impact on the server’s performance. Whether to use parallel or serial backups can depend on many factors, including system load, hardware configuration, how many tape drives are available, and so on.

20.12.2 To Create Backup Groups

A backup group is an arbitrary set of user mailboxes defined by regular expressions. By organizing user mailboxes into backup groups, you can define more flexible backup management.

For example, you could create three backup groups, the first containing user IDs starting with the letters A through L, the second with users whose user IDs begin with M-Z, and the third with users whose user IDs begin with a number. Administrators could use these backup groups to backup mailboxes in parallel, or perhaps only certain groups on one day and other groups on another.

There are several things to remember about backup groups:

1. These are arbitrary *virtual* groups of mail users. They do not precisely map to the message store directory ([Figure 20–1](#)), although it may look like it.
2. They are defined by the administrator using UNIX regular expressions.
3. The regular expressions are defined in the configuration file `msg-svr-base/config/backup-groups.conf`
4. When backup groups are referenced in `imsbackup` and `imsrestore`, they use the path format: `/partition_name/backup_group`

The format of `backup-groups.conf` is as follows:

```
group_name=definition  
group_name=definition
```

```
.
```

```
.
```

Using the example described in the paragraph above, the following definitions would be used to create the three backup groups:

```
groupA=[a-l].*  
groupB=[m,-z].*  
groupC=[0-9].*
```

You can now scope `imsbackup` and `ims restore` at several levels. You can backup/restore the whole message store using the `backup` command:

```
imsbackup -f device /
```

To backup all mailboxes for all users in `groupA` use the following:

```
imsbackup -f device /partition/groupA
```

The default partition is called `primary`.

20.12.2.1

Pre-defined Backup Group

Messaging Server includes one predefined backup group that is available without creating the `backup-groups` configuration file. This group is called `user`; it includes all users. For example, the following will backup all users on the `primary` partition:

```
imsbackup -f backupfile /primary/user
```

20.12.3

Messaging Server Backup and Restore Utilities

To back up and restore your data, Messaging Server provides the `imsbackup` and `ims restore` utilities. Note that the `imsbackup` and `ims restore` utilities do not have the advanced features found in general purpose tools like Legato Networker. For example, the utilities have only very limited support for tape auto-changers, and they cannot write a single store to multiple concurrent devices. Comprehensive backup will be achieved via plug-ins to generalized tools like Legato Networker. For more information about using Legato Networker, see “[20.12.6 To Use Legato Networker](#)” on page 653

20.12.3.1

The `imsbackup` Utility

With `imsbackup`, you can write selected contents of the message store to any serial device, including magnetic tape, a UNIX pipe, or a plain file. The backup or selected parts of the backup may later be recovered by using the `ims restore` utility. The output of `imsbackup` can be piped to `ims restore`.

The following example backs up the entire message store to /dev/rmt/0:

```
imsbackup -f /dev/rmt/0 /
```

This backs up the mailboxes of user ID joe to /dev/rmt/0:

```
imsbackup -f /dev/rmt/0 /primary/user/joe
```

This example backs up all the mailboxes of all the users defined in the backup group groupA to backupfile (see “[20.12.2 To Create Backup Groups](#)” on page 648):

```
imsbackup -f- /primary/groupA > backupfile
```

Incremental Backup

The following example will back up messages stored from May 1, 2004 at 1:10 pm to the present. The default is to backup all the messages regardless of their dates:

```
imsbackup -f /dev/rmt/0 -d 20040501:131000 /
```

This command uses the default blocking factor of 20. For a complete syntax description of the imsbackup command, see the [Sun Java System Messaging Server 6.3 Administration Reference](#).

20.12.3.2

The imsrestore Utility

To restore messages from the backup device, use the imsrestore command. For example, the following command restores messages for user1 from the file backupfile.

```
imsrestore -f backupfile /primary/user1
```

For a complete syntax description of the imsbackup command, see the [Sun Java System Messaging Server 6.3 Administration Reference](#).

20.12.4

Excluding Bulk Mail When You Perform Backups

When you perform a backup operation, you can specify mailboxes that will be excluded from being backed up. By excluding bulk or trash mailboxes that can accrue large numbers of unimportant messages, you can streamline the backup session, reduce the time to complete the operation, and minimize the disk space required to store the backup data.

To exclude mailboxes, specify a value for the configutil parameter local.store.backup.exclude.

You can specify a single mailbox or a list of mailboxes separated by the '%' character. ('%' is an illegal character in a mailbox name.) For example, you could specify the following values:

Trash

Trash%Bulk Mail%Third Class Mail

In the first example, the folder Trash is excluded. In the second example, the folders Trash, Bulk Mail, and Third Class Mail are excluded.

The backup utility backs up all folders in a user mailbox except those folders specified in the `local.store.backup.exclude` parameter.

This feature works with the Messaging Server backup utility, Legato Networker, and third-party backup software.

You can override the `local.store.backup.exclude` setting and back up an excluded mailbox by specifying its full logical name during the operation. Suppose the Trash folder has been excluded. You can still back up Trash by specifying, for example:

`/primary/user/user1/trash`

However, if you specify

`/primary/user/user1`

the Trash folder is excluded.

20.12.5 Considerations for Partial Restore

A partial restore is when only a part of the message store is restored. A full restore is when the entire message store is restored. The message store uses a single-copy message system. That is, only a single copy of any message is saved in the store as a single file. Any other instances of that message (like when a message is sent to multiple mailboxes) are stored as links to that copy. Because of this, there are implications when restoring messages. For example:

- **Full Restore.** During a full restore, linked messages will still point to the same inode as the message file to which they are linked.
- **Partial Backup/Restore.** During a partial backup and partial restore, however, the single-copy characteristic of the message store might not be preserved.

The following examples demonstrate what happens to a message used by multiple users when a partial restore is performed. Assume there are three messages, all the same, belonging to three users A, B, and C, as follows:

`A/INBOX/1`

`B/INBOX/1`

`C/INBOX/1`

Example 1. In the first example, the system performs a partial backup and full restore procedure as follows:

1. Back up mailboxes for users B and C.
2. Delete mailboxes of users B and C.
3. Restore the backup data from step 1.

In this example, B/INBOX/1 and C/INBOX/1 are assigned a new inode number and the message data is written to a new place on the disk. Only one message is restored; the second message is a hard link to the first message.

Example 2. In this example, the system performs a full backup and a partial restore as follows:

1. Perform full backup.
2. Delete mailboxes for user A.
3. Restore mailboxes for user A.

A/INBOX/1 is assigned a new inode number.

Example 3. In this example, partial restore might require more than one attempt:

1. Perform full backup.
B/INBOX/1 and C/INBOX/1 are backed up as links to A/INBOX/1.
2. Delete mailboxes for users A and B.
3. Restore mailboxes for user B.
The restore utilities ask the administrator to restore A/INBOX first.
4. Restore mailboxes for users A and B.
5. Delete mailboxes for user A (optional).

Note – If you want to ensure that all messages are restored for a partial restore, you can run the `imsbackup` command with the `-i` option. The `-i` option backs up every message multiple times if necessary.

If the backup device is seekable (example: a drive or tape), `imsrestore` seeks to the position containing A/INBOX/1 and restore it as B/INBOX/1. If the backup device is non-seekable (example: a UNIX pipe), `imsrestore` logs the object ID and the ID of the depending (linked) object to a file, and the administrator must invoke `imsrestore` again with the `-r` option to restore the missing message references.

20.12.5.1

To Restore Messages from a Mailbox that has Been Incrementally Backed-up

If you are restoring messages from a mailbox that has been incrementally backed-up, and if that mailbox exists on the server on which you wish to restore the messages, then restoring the

messages requires a straightforward `ims restore`. However, if you wish to restore messages from a mailbox that has been incrementally backed-up, and if that mailbox no longer exists, you must follow different restore procedures.

Use one of the following procedures to restore messages to a mailbox that does not exist on the message store server:

- During the restore operation, disable delivery of messages to the user. Do this by setting the LDAP attribute `mailDeliveryOption` to `hold`.
- Before you use `ims restore`, create the mailbox with the `mboxutil -c` command.

The reason why these instructions must be followed for restoring an incremental backup is as follows: When a mailbox has been deleted or is being migrated, the `ims restore` utility recreates the mailbox with the mailbox unique identification validity and message unique identifications (UIDs) stored in the backup archive.

In the past, when `ims restore` would recreate a deleted or migrated mailbox, it would assign a new UID validity to the mailbox and new UIDs to the messages. In that situation, a client with cached messages would have to resynchronize the mailbox UID validity and message UIDs. The client would have to download the new data again, increasing the workload on the server.

With the new `ims restore` behavior, the client cache remains synchronized, and the restore process operates transparently with no negative impact on performance.

If a mailbox exists, `ims restore` assigns new UIDs to the restored messages so that the new UIDs remain consistent with the UIDs already assigned to existing messages. To ensure UID consistency, `ims restore` locks the mailbox during the restore operation. However, because `ims restore` now uses the mailbox UID validity and message UIDs from the backup archive instead of assigning new UID values, UIDs could become inconsistent if you perform incremental backups and restores.

If you perform incremental backups with the `-d date` option of the `imsbackup` utility, you might have to invoke `ims restore` multiple times to complete the restore operation. If incremental backups were performed, you must restore the latest full backup and all subsequent incremental backups.

New messages can be delivered to the mailbox between the restore operations, but in this case, the message UIDs can become inconsistent. To prevent inconsistency in the UIDs, you need to take one of the actions described above.

20.12.6 To Use Legato Networker

Messaging Server includes a backup API that provides an interface with third-party backup tools, such as Legato Networker. The physical message store structure and data format are encapsulated within the backup API. The backup API interacts directly with the message store. It presents a logical view of the message store to the backup service. The backup service uses the conceptual representation of the message store to store and retrieve the backup objects.

Messaging Server provides an Application Specific Module (ASM) that can be invoked by the Legato Networker's save and recover commands to back up and restore the message store data. The ASM then invokes the Messaging Server `imsbackup` and `imsrestore` utilities.

Note – This section provides information about how to use Legato Networker with the Messaging Server message store. To understand the Legato Networker interface, see your Legato documentation.

▼ To Back Up Data Using Legato Networker

- 1 **Create a symbolic link from `/usr/lib/nsr/imsasm` to `msg-srv-base/lib/msg/imsasm`**
- 2 **From Sun or Legato, obtain a copy of the `nsrfile` binary and copy it to the following directory: `/usr/bin/nsr`**

Note that this is required only if you are using an older version of Networker (5.x). With Networker 6.0 and above, `nsrfile` is automatically installed under `/usr/bin/nsr`.
- 3 **If you want to back up users by groups, perform the following steps:**
 - a. **Create a backup group file as described in “[20.12.2 To Create Backup Groups](#)” on page 648**
 - b. **To verify your configuration, run `mkbackupdir.sh`.**

Look at the directory structure created by `mkbackupdir.sh`. The structure should look similar to that shown in [Table 20–4](#).

Note that if you do not specify a `backup-groups.conf` file, the backup process will use the default backup group `ALL` for all users.
- 4 **In the directory `/nsr/res/`, create a `res` file for your save group to invoke the `mkbackupdir.sh` script before the backup. See [Table 20–4](#) for an example.**

Note – Earlier versions of Legato Networker have a limitation of 64 characters for the save set name. If the name of this directory plus the logical name of the mailbox (for example, `/primary/groupA/fred`) is greater than 64 characters, then you must run `mkbackupdir.sh -p`. Therefore, you should use a short path name for the `-p` option of `mkbackupdir.sh`. For example the following command will create the backup image under the directory `/backup`:

```
mkbackupdir.sh -p /backup
```

Important: The backup directory must be writable by the message store owner (example: `mailsrv`).

Table 20–6 shows a sample backup groups directory structure.

```
/backup/primary/groupA/amy
    /bob
    /carly
    /groupB/mary
        /nancy
        /zelda
    /groupC/123go
        /lbill
        /354hut
```

The example below shows a sample res file named `IMS.res` in the `/nsr/res` directory:

```
type: savenpc;
precmd: "echo mkbackupdir started",
    "/usr/siroe/server5/msg-siroe/bin/mkbackupdir.sh -p /backup";
pstcmd: "echo imsbackup Completed";
timeout: "12:00 pm";
```

You are now ready to run the Legato Networker interface as follows:

- 5 Create the Messaging Server save group if necessary.**
 - a. Run `nwadmin`.
 - b. Select **Customize | Group | Create**.
- 6 Create a backup client using `savenpc` as the backup command:**
 - a. Set the save set to the directory created by `mkbackupdir`.

For a single session backup, use `/backup`

For parallel backups, use `/backup/server/group`

Be sure you've already created `group` as defined in “20.12.2 To Create Backup Groups” on page 648

You must also set the parallelism to the number of backup sessions.

See “To Back Up Data Using Legato Networker” on page 654.
- 7 Select Group Control | Start to test your backup configuration.**

Example. Creating A Backup Client in Networker:

To create a backup client in Networker. From nwadmin, select Client | Client Setup | Create

```
Name: siroe  
Group: IMS  
Savesets:/backup/primary/groupA  
/backup/secondary/groupB  
/backup/tertiary/groupC
```

```
.
```



```
Backup Command:savepnpc  
Parallelism: 4
```

20.12.6.1

Restoring Data Using Legato Networker

To recover data, you can use the Legato Networker nwrecover interface or the recover command-line utility. The following example recovers user a1's INBOX:

```
recover -a -f -s siroe /backup/siroe/groupA/a1/INBOX
```

The next example recovers the entire message store:

```
recover -a -f -s siroe /backup/siroe
```

20.12.7

To Use a Third Party Backup Software (Besides Legato)

Messaging Server provides two message store backup solutions, the command line imsbackup and the Solstice Backup (Legato Networker). A large message store running a single imbackup to backup the entire message store can take a significant amount of time. The Legato solution supports concurrent backup sessions on multiple backup devices. Concurrent backup can shorten backup time dramatically (backups of 25GB of data per hour have been achieved).

If you are using another third party concurrent backup software (for example, Netbackup), you may use the following method to integrate your backup software with the Messaging Server.

▼ To Use a Third Party Backup Software (Besides Legato)

- 1 Divide your users into groups (see “[20.12.2 To Create Backup Groups](#)” on page 648) and create a `backup-groups.conf` file under the directory `msg-svr-base/config/`.

Note – This backup solution requires additional disk space. To backup all the groups concurrently, the disk space requirement is 2 times the message store size. If you do not have that much disk space, divide your users into smaller groups, and then backup a set of groups at a time. For example group1 - group5, group6 - group10. Remove the group data files after backup.

2 Run `imsbackup` to backup each group into files under a staging area.

The command is `imsbackup -f <device> /<instance>/<group>`

You can run multiple `imsbackup` processes simultaneously. For example:

```
# imsbackup -f- /primary/groupA > /bkdata/groupA &
# imsbackup -f- /primary/groupB > /bkdata/groupB &
...

```

`imsbackup` does not support large files, if the backup data is larger than 2 GB, you need to use the `-f-` option to write the data to `stdout` and then pipe the output to a file.

3 Use your third party backup software to backup the group data files in the staging area (in our example that is `/bkdata`).

4 To restore a user, identify the group filename of the user, restore that file from tape, and then use `imsrestore` to restore the user from the data file.

Note that `imsrestore` does not support large files. If the data file is larger than 2GB. Use the following command:

```
# cat /bkdata/groupA | imsrestore -f- /primary/groupA/andy
```

20.12.8 Troubleshooting Backup and Restore Problems

This section describes common backup and restore problems and their solutions.

- **Problem:** `msprobe` restarts everything during a long `imsrestore` during message store migration. This can also happen with `imsbackup`, `imsimport`, or any processing intensive utility.
- **Solution:** When `imsrestore` or any processing intensive operation takes significantly more system resources than normal, and continues doing so longer than the `msprobe` interval, there may be a temporary backlog of DB transaction log files to be cleared. If there are more files than specified in `local.store.maxlog`, then `msprobe` may erroneously restart all the processes during a restore. To prevent this from happening, disable `msprobe` during the `imsrestore`.
- **Problem:** When I do an restore of a folder or INBOX using `imsrestore` or `imsasm`, it appends all the messages in that folder onto the current folder. This results in multiple copies of the messages in that folder.

Solution: Make sure the `-i` flag of `ims restore` is not set in the `imsasm` script.

- **Problem:** I want to do an incremental backup of just new messages added in a mail folder, but when I try, the entire folder gets backed up. How do I just back up the new messages?

Solution: Set the `-d datetime` flag on `imsbackup`. This will backup messages stored from the specified date and time to the present. The default is to backup all messages regardless of their dates.

20.12.9 Message Store Disaster Backup and Recovery

A disaster refers to a catastrophic failure of the entire message store as opposed to a mailbox or set of mailboxes. That is, a situation where all data on the message store servers are lost. A complete message store disaster restore will consist of restoring the following lost data:

- All message store data. These can be backed up using the procedures described in “[20.12 Backing Up and Restoring the Message Store](#)” on page 646. If file system backup method is used, be sure to backup the following data:
 - All message store partitions
 - The message store database files at `msg-svr-base/data/store/mboxlist`.
 - The message store database snapshots at
`msg-svr-base/data/store/dbdata/snapshots` (Note that the location of message store database snapshot files can be configured with the configutil parameter `local.store.snapshotpath`.)
- Configuration data. Including the local configuration file at `msg-svr-base/data/config`

If you want to backup your Message Store for disaster recovery, you can use a file system snapshot tools to take a snapshot of the file system. The snapshot **must** be a *point-in-time* file system snapshot. Otherwise, the `mboxlist` backup cannot be used (the `mboxlist` database must be restored from a complete database snapshot).

It is best to capture all the data (message store partitions, database files and so on) at the same point-in-time, however, if this cannot be done, then you must backup the data in this order:

1. Database snapshots
2. Database files
3. Message store partitions
4. Configuration data

If the message store partitions and the database files are not backed up with the same point-in-time snapshot, run `reconstruct -m` after restoring the file system snapshots. This will synchronize the database and the store partitions.

20.13 Monitoring User Access

Messaging Server provides the command, `imsconnutil`, which allows you to monitor user's message store access via IMAP, POP and http. You can also determine the last log in and log out of users. This command works on a per message store basis and will not work across message stores.

Note – Use of this function or other Messaging Server functions to monitor, read or otherwise access user's email may constitute a potential source of liability if used in violation of applicable laws or regulations or if used in violation of the customer's own policies or agreements.

This command requires root access by the system user (default: `mailsrv`), and you must set the configuration variables `local imap.enableuserlist`, `local http.enableuserlist`, `local.enablelastaccess` to 1.

To list users currently logged on via IMAP or any web mail client, use the following command:

```
# imsconnutil -c
```

To list the last IMAP, POP, or Messenger Express access (log in and log out) of every user on the message store use:

```
# imsconnutil -a
```

The following command does two things: 1) it determines whether the specified user is currently logged on via IMAP or Messenger Express or any client that connects via mshttp (note that this does not work for POP because POP users generally do not stay connected), and 2) it lists the last time the users have logged on and off:

```
# imsconnutil -c -a -u user_ID
```

Note that a list of users can be input from a file, one user per line, using the following command:

```
# imsconnutil -c -a -f filename
```

You can also specify a particular service (imap or http) using the `-s` flag. For example, to list whether a particular user ID is logged onto IMAP or not, use the following command:

```
# imsconnutil -c -s imap -u user_ID
```

Note that the `-k` option may only work if IMAP IDLE is configured. For a complete description of the `imsconnutil` syntax, refer to “[imsconnutil](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

Here is some example output:

```
$ ./imsconnutil -a -u soroork

      UID      IMAP last access      HTTP last access      POP last access
=====
ed    08/Jul/2003:10:49:05    10/Jul/2003:14:55:52  ---NOT-RECORDED---

$ ./imsconnutil -c

IMAP
UID      TIME                  AUTH          TO                      FROM
=====
ed    17/Jun/2003:11:24:03  plain        172.58.73.45:193  129.157.12.73:2631
bil   17/Jun/2003:04:28:43  plain        172.58.73.45:193  129.158.16.34:2340
mia   17/Jun/2003:09:36:54  plain        172.58.73.45:193  192.18.184.103:3744
jay   17/Jun/2003:05:38:46  plain        172.58.73.45:193  129.159.18.123:3687
pau   17/Jun/2003:12:23:28  plaintext   172.58.73.45:193  192.18.194.83:2943
ton   17/Jun/2003:05:38:46  plain        172.58.73.45:193  129.152.18.123:3688
ani   17/Jun/2003:12:26:40  plaintext   172.58.73.45:193  192.18.164.17:1767
ani   17/Jun/2003:12:25:17  plaintext   172.58.73.45:193  129.150.17.34:3117
jac   17/Jun/2003:12:26:32  plaintext   172.58.73.45:193  129.150.17.34:3119
ton   17/Jun/2003:12:25:32  plaintext   172.58.73.45:193  192.18.148.17:1764
=====
10 users were logged in to imap.
Feature is not enabled for http.
-----
```

20.14 Troubleshooting the Message Store

This section provides guidelines for actively maintaining your message store. In addition, this section describes other message store recovery procedures you can use if the message store becomes corrupted or unexpectedly shuts down. Note that the section on these additional message store recovery procedures is an extension of “[20.14.3 Repairing Mailboxes and the Mailboxes Database](#)” on page 667.

Prior to reading this section, it is strongly recommended that you review this chapter as well as the command-line utility and configutil chapters in the Sun Java System Messaging Server Administration Reference. Topics covered in this section include:

- “[20.14.1 Standard Message Store Monitoring Procedures](#)” on page 661
- “[20.14.2 Message Store Startup and Recovery](#)” on page 664
- “[20.14.3 Repairing Mailboxes and the Mailboxes Database](#)” on page 667
- “[20.14.4 Common Problems and Solutions](#)” on page 671

20.14.1 Standard Message Store Monitoring Procedures

This section outlines standard monitoring procedures for the message store. These procedures are helpful for general message store checks, testing, and standard maintenance.

For additional information, see “[27.7 Monitoring the Message Store](#)” on page [877](#).

20.14.1.1

Check Hardware Space

A message store should have enough additional disk space and hardware resources. When the message store is near the maximum limit of disk space and hardware space, problems might occur within the message store.

Inadequate disk space is one of the most common causes of the mail server problems and failure. Without space to write to the message store, the mail server will fail. In addition, when the available disk space goes below a certain threshold, there will be problems related to message delivery, logging, and so forth. Disk space can be rapidly depleted when the clean up function of the stored process fails and deleted messages are not expunged from the message store.

For information on monitoring disk space, see “[20.11.5 To Monitor Disk Space](#)” on page [642](#) and “[27.7 Monitoring the Message Store](#)” on page [877](#).

20.14.1.2

Check Log Files

Check the log files to make sure the message store processes are running as configured. Messaging Server creates a separate set of log files for each of the major protocols, or services, it supports: SMTP, IMAP, POP, and HTTP. You can look at the log files in the directory *msg-svr-base/log/*. You should monitor the log files on a routine basis.

Be aware that logging can impact server performance. The more verbose the logging you specify, the more disk space your log files will occupy for a given amount of time. You should define effective but realistic log rotation, expiration, and backup policies for your server. For information about defining logging policies for your server, see [Chapter 25, “Managing Logging.”](#)

20.14.1.3

Check User IMAP/POP/Webmail Session by Using Telemetry

Messaging Server provides a feature called telemetry that can capture a user’s entire IMAP, POP or HTTP session into a file. This feature is useful for debugging client problems. For example, if a user complains that their message access client is not working as expected, this feature can be used to trace the interaction between the access client and Messaging Server.

To capture a POP session, create the following directory:

msg-svr-base/data/telemetry/pop_or_imap_or_http/userid

To capture a POP session, create the following directory:

msg-svr-base/data/telemetry/pop/userid

To capture an IMAP session, create the following directory:

msg-svr-base/data/telemetry/imap/userid

To capture a Webmail session, create the following directory:

msg-svr-base/data/telemetry/http/userid

Note that the directory must be owned or writable by the messaging server userid.

Messaging Server will create one file per session in that directory. Example output is shown below.

```
LOGIN redb 2003/11/26 13:03:21
>0.017>1 OK User logged in
<0.047<2 XSERVERINFO MANAGEACCOUNTURL MANAGELISTSURL MANAGEFILTERSURL
>0.003>* XSERVERINFO MANAGEACCOUNTURL {67}
http://redb@cuisine.blue.planet.com:800/bin/user/admin/bin/enduser
MANAGELISTSURL NIL MANAGEFILTERSURL NIL
2 OK Completed
<0.046<3 select "INBOX"
>0.236>* FLAGS (\Answered flagged draft deleted \Seen $MDNSent Junk)
* OK [PERMANENTFLAGS (\Answered flag draft deleted \Seen $MDNSent Junk \*)]
* 1538 EXISTS
* 0 RECENT
* OK [UNSEEN 23]
* OK [UIDVALIDITY 1046219200]
* OK [UIDNEXT 1968]
3 OK [READ-WRITE] Completed
<0.045<4 UID fetch 1:* (FLAGS)
>0.117>* 1 FETCH (FLAGS (\Seen) UID 330)
* 2 FETCH (FLAGS (\Seen) UID 331)
* 3 FETCH (FLAGS (\Seen) UID 332)
* 4 FETCH (FLAGS (\Seen) UID 333)
* 5 FETCH (FLAGS (\Seen) UID 334)
<etc>
```

To disable the telemetry logging, move or remove the directory that you created.

20.14.1.4

Check stored Processes

The `stored` function performs a variety of important tasks such as deadlock and transaction operations of the message database, enforcing aging policies, and expunging and erasing messages stored on disk. If `stored` stops running, Messaging Server will eventually run into problems. If `stored` doesn't start when `start-msg` is run, no other processes will start.

- Check that the stored process is running. Run `imcheck`
- Check for the log file build up in `store_root/mboxlist`.
- Check for stored messages in the default log file `msg-svr-base/log/default/default`
- Check that the time stamps of the following files (in directory `msg-svr-base/config/`) are updated whenever one of the following functions are attempted by the stored process:

TABLE 20–12 stored Operations

stored Operation	Function
<code>stored.ckpt</code>	Touched when a database checkpoint was initiated. Stamped approximately every 1 minute.
<code>stored.lcu</code>	Touched at every database log cleanup. Time stamped approximately every 5 minutes.
<code>stored.per</code>	Touched at every spawn of peruser db write out. Time stamped once an hour.

For more information on the stored process, see “[20.11.6 The stored Daemon](#)” on page 642 chapter of the *Sun Java System Messaging Server 6.3 Administration Reference*.

For additional information on monitoring the stored function, see “[27.7 Monitoring the Message Store](#)” on page 877

20.14.1.5

Check Database Log Files

Database log files refer to sleepycat transaction checkpointing log files (in directory `store_root/mboxlist`). If log files accumulate, then database checkpointing is not occurring. In general, there are two or three database log files during a single period of time. If there are more files, it could be a sign of a problem.

20.14.1.6

Check User Folders

If you want to check the user folders, you might run the command `reconstruct -r -n` (recursive no fix) which will review any user folder and report errors. For more information on the reconstruct command, see “[20.14.3 Repairing Mailboxes and the Mailboxes Database](#)” on page 667

20.14.1.7

Check for Core Files

Core files only exist when processes have unexpectedly terminated. It is important to review these files, particularly when you see a problem in the message store. On Solaris, use `coreadm` to configure core file location.

20.14.2 Message Store Startup and Recovery

Message store data consists of the messages, index data, and the message store database. While this data is fairly robust, on rare occasions there may be message store data problems in the system. These problems will be indicated in the default log file, and almost always will be fixed transparently. In rare cases an error message in the log file may indicate that you need to run the reconstruct utility. In addition, as a last resort, messages are protected by the backup and restore processes described in “[20.12 Backing Up and Restoring the Message Store](#)” on [page 646](#). This section will focus on the automatic startup and recovery process of `stored`.

The message store automates many recovery operations which were previously the responsibility of the administrator. These operations are performed by message store daemon `stored` during startup and include database snapshots and automatic fast recovery as necessary. `stored` thoroughly checks the message store’s database and automatically initiates repairs if it detects a problem.

`stored` also provides a comprehensive analysis of the state of the database via status messages to the default log, reporting on repairs done to the message store and automatic attempts to bring it into operation.

20.14.2.1

Automatic Startup and Recovery—Theory of Operations

The `stored` daemon starts before the other message store processes. It initializes and, if necessary, recovers the message store database. The message store database keeps folder, quota, subscription, and message flag information. The database is logging and transactional, so recovery is already built in. In addition, some database information is copied redundantly in the message index area for each folder.

Although the database is fairly robust, on the rare occasions that it breaks, in most cases `stored` recovers and repairs it transparently. However, whenever `stored` is restarted, you should check the default log files to make sure that additional administrative intervention is not required. Status messages in the log file will tell you to run `reconstruct` if the database requires further rebuilding.

Before opening the message store database, `stored` analyzes its integrity and sends status messages to the default log under the category of *warning*. Some messages will be useful to administrators and some messages will consist of coded data to be used for internal analysis. If `stored` detects any problems, it will attempt to fix the database and try starting it again.

When the database is opened, `stored` will signal that the rest of the services may start. If the automatic fixes failed, messages in the default log will specify what actions to take. See “[Error Messages Signifying that `reconstruct` is Needed](#)” on [page 665](#)

In previous releases, `stored` could start a recovery process which would take a very long time leaving the administrator wondering if `stored` was “stuck.” This type of long recovery has been

removed and stored should determine a final state in less than a minute. However, if stored needs to employ recovery techniques such as recovering from a snapshot, the process may take a few minutes.

After most recoveries, the database will usually be up-to-date and nothing else will be required. However, some recoveries will require a reconstruct -m in order to synchronize redundant data in the message store. Again, this will be stated in the default log, so it is important to monitor the default log after a startup. Even though the message store will seem to be up and running normally, it is important to run any requested operations such as reconstruct.

Another reason for reading the log file is to determine what caused damage to the database in the first place. Although stored is designed to bring up the message store regardless of any problem on the system, you will still want to try to ascertain cause of the database damage as this may be a sign of a larger hidden problem.

Error Messages Signifying that reconstruct is Needed

This section describes the type of error messages that require reconstruct to be run.

When the error message indicates mailbox error, run reconstruct <mailbox>. Example:

```
"Invalid cache data for msg 102 in mailbox user/joe/INBOX. Needs reconstruct"  
"Mailbox corrupted, missing fixed headers: user/joe/INBOX"  
"Mailbox corrupted, start_offset beyond EOF: user/joe/INBOX"
```

When the error message indicates a database error, run reconstruct -m. Example:

```
"Removing extra database logs. Run reconstruct -m soon after startup to resync  
redundant data"  
"Recovering database from snapshot. Run reconstruct -m soon after startup to  
resync redundant data"
```

Database Snapshots

A snapshot is a hot backup of the database and is used by stored to restore a broken database transparently in a few minutes. This is much quicker than using reconstruct, which relies on the redundant information stored in other areas.

Message Store Database Snapshot—Theory of Operations

Snapshots of the database, located in the mboxlist directory, are taken automatically, by default, once every 24 hours. Snapshots are copied by default into a subdirectory of the store directory. By default, there are five snapshots kept at any given time: one live database, three snapshots, and one database/removed copy. The database/removed copy is newer and is an emergency copy of the database which is put into a subdirectory removed of the mboxlist database directory.

If the recovery process decides to remove the current database because it is determined to be bad, `stored` will move it into the `removed` directory if it can. This allows the database to be analyzed if desired.

The data move will only happen once a week. If there is already a copy of the database there, `stored` will not replace it every time the store comes up. It will only replace it if the data in the removed directory is older than a week. This will prevent the original database which had the problem from being replaced too soon by successive startups.

To Specify Message Store Database Snapshot Interval and Location

There should be five times as much space for the database and snapshots combined. It is highly recommended that the administrator reconfigure snapshots to run on a separate disk, and that it is tuned to the system's needs.

If `stored` detects a problem with the database on startup, the best snapshot will automatically be recovered. Three snapshot variables can set the following parameters: the location of the snapshot file, the interval for taking snapshots, number of snapshots saved. These configutil parameters are shown in [Table 20–13](#).

Having a snapshot interval which is too small will result in a frequent burden to the system and a greater chance that a problem in the database will be copied as a snapshot. Having a snapshot interval too large can create a situation where the database will hold the state it had back when the snapshot was taken.

A snapshot interval of a day is recommended and a week or more of snapshots can be useful if a problem remains on the system for a number of days and you wish to go back to a period prior to the point at which the problem existed.

`stored` monitors the database and is intelligent enough to refuse the latest snapshot if it suspects the database is not perfect. It will instead retrieve the latest most reliable snapshot. Despite the fact that a snapshot may be retrieved from a day ago, the system will use more up to date redundant data and override the older snapshot data, if available.

Thus, the ultimate role the snapshot plays is to get the system as close to up-to-date and ease the burden of the rest of the system trying to rebuild the data on the fly.

TABLE 20–13 Message Store Database Snapshot Parameters

Parameter	Description
<code>local.store.snapshotpath</code>	Location of message store database snapshot files. Either existing absolute path or path relative to the <code>store</code> directory. Default: <code>dbdata/snapshots</code>

TABLE 20-13 Message Store Database Snapshot Parameters *(Continued)*

Parameter	Description
<code>local.store.snapshotinterval</code>	Minutes between snapshots. Valid values: 1 - 46080 Default: 1440 (1440 minutes = 1 day)
<code>local.store.snapshotdirs</code>	Number of different snapshots kept. Valid values: 2 - 367 Default: 3

20.14.3 Repairing Mailboxes and the Mailboxes Database

If one or more mailboxes become corrupt, you can use the `reconstruct` utility to rebuild the mailboxes or the mailbox database and repair any inconsistencies.

The `reconstruct` utility rebuilds one or more mailboxes, or the master mailbox file, and repairs any inconsistencies. You can use this utility to recover from almost any form of data corruption in the mail store. See “[Error Messages Signifying that reconstruct is Needed](#)” on page 665

[Table 20–14](#) lists the `reconstruct` options. For detailed syntax and usage requirements, see the “`reconstruct`” in *Sun Java System Messaging Server 6.3 Administration Reference*.

TABLE 20-14 `reconstruct` Options

Option	Description
<code>-e</code>	Removes the <code>store.exp</code> file before reconstructing. This eliminates any internal store record of removed messages which have not been cleaned out by the store process. It would also be useful to use the <code>-f</code> option when using <code>-i</code> or <code>-e</code> , because these options only work if the folder is actually reconstructed. Similarly, if you use the <code>-n</code> option (which performs a check, not a reconstruction), the <code>-i</code> and <code>-e</code> options do not work. Running a <code>reconstruct -e</code> will not recover removed messages if <code>reconstruct</code> does not detect damage. An <code>-f</code> will force the <code>reconstruct</code> .
<code>-i</code>	Sets the <code>store.idx</code> file length to zero before reconstructing. It would also be useful to use the <code>-f</code> option when using <code>-i</code> or <code>-e</code> , because these options only work if the folder is actually reconstructed. Similarly, if you use the <code>-n</code> option (which performs a check, not a reconstruction), the <code>-i</code> and <code>-e</code> options do not work.
<code>-f</code>	Forces <code>reconstruct</code> to perform a fix on the mailbox or mailboxes.
<code>-l</code>	Reconstructs <code>lright.db</code> .
<code>-m</code>	Performs a consistency check and, if needed, repairs the mailboxes database. This option examines every mailbox it finds in the spool area, adding or removing entries from the mailbox's database as appropriate. The utility prints a message to the standard output file whenever it adds or removes an entry from the database. Specifically it fixes <code>folder.db</code> , <code>quota.db</code> , and <code>lright.db</code>

TABLE 20-14 reconstruct Options (Continued)

Option	Description
-n	Checks the message store only, without performing a fix on the mailbox or mailboxes. The -n option cannot be used by itself unless a mailbox name is provided. When a mailbox name is not provided, the -n option must be used with the -r option. The -r option may be combined with the -p option. For example, any of the following commands are valid: reconstruct -n user/dulcinea/INBOX reconstruct -n -r reconstruct -n -r -p primary reconstruct -n -r user/dulcinea/
-o	Obsolete, see <code> mboxutil -o</code>
-o -d <i>filename</i>	Obsolete, see <code> mboxutil -o</code>
-p <i>partition</i>	The -p option is used with the -m option and limits the scope of the reconstruction to the specified partition. If the -p option is not specified, <code>reconstruct</code> defaults to all partitions. Specifically it fixes <code>folder.db</code> and, <code>quota.db</code> , but not <code>lright.db</code> . This is because fixing the <code>lright.db</code> requires scanning the acls for every user in the message store. Performing this for every partition is not very efficient. To fix <code>lright.db</code> run <code>reconstruct -l</code> . Specify a partition name; do not use a full path name.
-q	Fixes any inconsistencies in the quota subsystem, such as mailboxes with the wrong quota root or quota roots with the wrong quota usage reported. The -q option can be run while other server processes are running.
-r [mailto]	Repairs and performs a consistency check of the partition area of the specified mailbox or mailboxes. The -r option also repairs all sub-mailboxes within the specified mailbox. If you specify -r with no mailbox argument, the utility repairs the spool areas of all mailboxes within the user partition directory.
-u <i>user</i>	The -u option is used with the -m option and limits the scope of the reconstruction to the specified user. The -u option must be used with the -p option. If the -u option is not specified, <code>reconstruct</code> defaults to all partitions or to the partition specified with the -p option. Specify a user name; do not use a full path name.

20.14.3.1

To Rebuild Mailboxes

To rebuild mailboxes, use the -r option. You should use this option when:

- Accessing a mailbox returns one of the following errors: “System I/O error” or “Mailbox has an invalid format”.
- Accessing a mailbox causes the server to crash.
- Files have been added to or removed from the spool directory.

`reconstruct -r` first runs a consistency check. It reports any inconsistencies and rebuilds only if it detects any problems. Consequently, performance of the `reconstruct` utility is improved with this release.

You can use `reconstruct` as described in the following examples:

To rebuild the spool area for the mailboxes belonging to the user `daphne`, use the following command:

```
reconstruct -r user/daphne
```

To rebuild the spool area for all mailboxes listed in the mailbox database:

```
reconstruct -r
```

You must use this option with caution, however, because rebuilding the spool area for all mailboxes listed in the mailbox database can take a very long time for large message stores. (See “[20.14.3.3 reconstruct Performance](#)” on page 670.) A better method for failure recovery might be to use multiple disks for the store. If one disk goes down, the entire store does not. If a disk becomes corrupt, you need only rebuild a portion of the store by using the `-p` option as follows:

```
reconstruct -r -p subpartition
```

To rebuild mailboxes listed in the command-line argument only if they are in the `primary` partition:

```
reconstruct -p primary mbox1 mbox2 mbox3
```

If you do need to rebuild all mailboxes in the `primary` partition:

```
reconstruct -r -p primary
```

If you want to force `reconstruct` to rebuild a folder without performing a consistency check, use the `-f` option. For example, the following command forces a `reconstruct` of the user folder `daphne`:

```
reconstruct -f -r user/daphne
```

To check all mailboxes without fixing them, use the `-n` option as follows:

```
reconstruct -r -n
```

20.14.3.2 Checking and Repairing Mailboxes

To perform a high-level consistency check and repair of the mailboxes database:

```
reconstruct -m
```

To perform a consistency check and repair of the primary partition:

```
reconstruct -p primary -m
```

Note – Running reconstruct with the -P and -m flags together will not fix lright.db. This is because fixing the lright.db requires scanning the ACLs for every user in the message store. Performing this for every partition is not very efficient. To fix the lright.db run reconstruct -l

To perform a consistency check and repair of an individual user's mailbox named john:

```
reconstruct -p primary -u john -m
```

You should use the -m option when:

- One or more directories were removed from the store spool area, so the mailbox database entries also need to be removed.
- One or more directories were restored to the store spool area, so the mailbox database entries also need to be added.
- The stored -d option is unable to make the database consistent.

If the stored -d option is unable to make the database consistent, you should perform the following steps in the order indicated:

- Shut down all servers.
- Remove all files in *store_root/mboxlist*.
- Restart the server processes.
- Run reconstruct -m to build a new mailboxes database from the contents of the spool area.

20.14.3.3

reconstruct Performance

The time it takes reconstruct to perform an operation depends on the following factors:

- The kind of operation being performed and the options chosen
- Disk performance
- The number of folders when running reconstruct -m
- The number of messages when running reconstruct -r
- The overall size of the message store
- What other processes the system is running and how busy the system is
- Whether or not there is ongoing POP, IMAP, HTTP, or SMTP activity. Note that reconstruct is designed to run with the store services up. It is not necessary to keep the store offline to run reconstruct.

The reconstruct -r option performs an initial consistency check; this check improves reconstruct performance depending on how many folders must be rebuilt.

The following performance was found with a system with approximately 2400 users, a message store of 85GB, and concurrent POP, IMAP, or SMTP activity on the server:

- `reconstruct -m` took about 1 hour
- `reconstruct -r -f` took about 18 hours

Note – A reconstruct operation may take significantly less time if the server is not performing ongoing POP, IMAP, HTTP, or SMTP activity.

20.14.4 Common Problems and Solutions

This section lists common message store problems and solutions:

- “20.14.4.1 Reduced Message Store Performance” on page 671
- “20.14.4.2 Linux - Messaging Server Patch 120230-08 IMAP, POP and HTTP Servers Not Starting Due to Over Sessions Per Process” on page 671
- “20.14.4.3 Messenger Express or Communications Express Not Loading Mail Page” on page 672
- “20.14.4.4 Command Using Wildcard Pattern Doesn’t Work” on page 672
- “20.14.4.5 Unknown/invalid Partition” on page 673
- “20.14.4.6 User Mailbox Directory Problems” on page 673
- “20.14.4.7 Store Daemon Not Starting” on page 674
- “20.14.4.8 User Mail Not Delivered Due to Mailbox Overflow” on page 674
- “20.14.4.9 IMAP Events Become Slow” on page 675

20.14.4.1 Reduced Message Store Performance

Message store problems can occur if the `mboxlist` database cache is too small. Specifically, Message store performance can slow to unacceptable levels and can even dump core. Refer to **Tuning the `mboxlist` Database Cache** (<http://wikis.sun.com/display/CommSuite/Tuning+the+mboxlist+Database+Cache>).

20.14.4.2 Linux - Messaging Server Patch 120230-08 IMAP, POP and HTTP Servers Not Starting Due to Over Sessions Per Process

After installing this patch, when you try to start Messaging Server, the IMAP, POP and HTTP servers do not start and may send the following example error logs:

```
http server - log:  
[29/May/2006:17:44:37 +051800] usg197 httpd[6751]: General Critical: Not enough file  
descriptors to support 6000 sessions per process; Recommend ulimit -n 12851 or 87  
sessions per process.
```

```
pop server - log:
```

```
[29/May/2006:17:44:37 +051800] usg197 popd[6749]: General Critical: Not enough file descriptors to support 600 sessions per process; Recommend ulimit -n 2651 or 58 sessions per process.
```

Once these values setting in /opt/sun/messaging/sbin/configutil then imap server failed to start

```
imap server - log:  
[29/May/2006:17:44:37 +051800] usg197 imapd[6747]: General Critical: Not enough file descriptors to support 4000 sessions per process; Recommend ulimit -n 12851 or 58 sessions per process.
```

Set the appropriate number of file descriptors for all three server sessions. Additional file descriptors are available by adding a line similar to the following to /etc/sysctl.conf and using sysctl -p to reread that file:

```
fs.file-max = 65536
```

You must also add a line like the following to /etc/security/limits.conf:

```
* soft  nofile  65536  
* hard  nofile  65536
```

20.14.4.3

Messenger Express or Communications Express Not Loading Mail Page

If the user cannot load any Messenger Express pages or the Communications Express mail page, the problem may be that the data is getting corrupted after compression. This can sometimes happen if the system has deployed a outdated proxy server. To solve this problem, try setting local.service.http.gzip.static and local.service.http.gzip.dynamic to 0 to disable data compression. If this solves the problem, you may want to update the proxy server.

20.14.4.4

Command Using Wildcard Pattern Doesn't Work

Some UNIX shells may require quotes around wildcard parameters and some will not. For example, the C shell tries to expand arguments containing wild cards (*, ?) as files and will fail if no match is found. These pattern matching arguments may need to be enclosed in quotes to be passed to commands like mboxutil.

For example:

```
mboxutil -l -p user/usr44*
```

will work in the Bourne shell, but will fail with tsch and the C shell. These shells would require the following:

```
mboxutil -l -p "user/usr44"
```

If a command using a wildcard pattern doesn't work, verify whether or not you need to use quotes around wildcards for that shell.

20.14.4.5

Unknown/invalid Partition

A user can get the message “Unknown/invalid partition” in Messenger Express if their mailbox was moved to a new partition which was just created and Messaging Server was not refreshed or restarted. This problem only occurs on new partitions. If you now add additional user mailboxes to this new partition, you will not have to do a refresh/restart of Messaging Server.

20.14.4.6

User Mailbox Directory Problems

A user mailbox problem exists when the damage to the message store is limited to a small number of users and there is no global damage to the system. The following guidelines suggest a process for identifying, analyzing, and resolving a user mailbox directory problem:

1. Review the log files, the error messages, or any unusual behavior that the user observes.
2. To keep debugging information and history, copy the entire `store_root/mboxlist/` user directory to another location outside the message store.
3. To find the user folder that might be causing the problem, run the command `reconstruct -r -n`. If you are unable to find the folder using `reconstruct`, the folder might not exist in the `folder.db`.

If you are unable to find the folder using the `reconstruct -r -n` command, use the `hashdir` command to determine the location. For more information on `hashdir`, see “[20.11.2.3 The hashdir Utility](#)” on page 640 and the `hashdir` utility in the Messaging Server Command-line Utilities chapter of the *Sun Java System Messaging Server 6.3 Administration Reference*.

4. Once you find the folder, examine the files, check permissions, and verify the proper file sizes.
5. Use `reconstruct -r` (without the `-n` option) to rebuild the mailbox.
6. If `reconstruct` does not detect a problem that you observe, you can force the reconstruction of your mail folders by using the `reconstruct -r -f` command.
7. If the folder does not exist in the `mboxlist` directory (`store_root/mboxlist`), but exists in the `partition` directory `store_root/partition`), there might be a global inconsistency. In this case, you should run the `reconstruct -m` command.
8. If the previous steps do not work, you can remove the `store.idx` file and run the `reconstruct` command again.



Caution – You should only remove the `store.idx` file if you are sure there is a problem in the file that the `reconstruct` command is unable to find.

9. If the issue is limited to a problematic message, you should copy the message file to another location outside of the message store and run the command `reconstruct -r` on the `mailbox/` directory.

10. If you determine the folder exists on the disk (*store_root/partition* directory), but is apparently not in the database (*store_root/mboxlist* directory), run the command `reconstruct -m` to ensure message store consistency.

For more information on the `reconstruct` command, see “[20.14.3 Repairing Mailboxes and the Mailboxes Database](#)” on page [667](#)

20.14.4.7 Store Daemon Not Starting

If `stored` won’t start and returns the following error message:

```
# msg-svr-base/sbin/start-msg  
  
msg-svr-base: Starting STORE daemon ...Fatal error: Cannot  
find group in name service
```

This indicates that the UNIX group configured in `local.servergid` cannot be found. `Stored` and others need to set their `gid` to that group. Sometimes the group defined by `local.servergid` gets inadvertently deleted. In this case, create the deleted group, add `mailsrv` to the group, change ownership of the `instance_root` and its files to `mailsrv` and the group.

20.14.4.8 User Mail Not Delivered Due to Mailbox Overflow

The message store has a hard limit of two gigabytes for a `store.idx` file, which is equivalent to about one million messages in a single mailbox (folder). If a mailbox grows to the point that the `store.idx` file will attempt to exceed two gigabytes, the user will stop receiving any new email. In addition, other processes that handle that mailbox, such as `imapd`, `popd`, `mshttpd`, could also experience degraded performance.

If this problem arises, you will see errors in `mail.log_current` such as this:

```
05-Oct-2005 16:09:09.63 ims-ms Q 7 ... System I/O error. Administrator, check  
server log for details. System I/O error.
```

In addition, the MTA log file will have an errors such as this:

```
[05/Oct/2005:16:09:09 +0900] jmail ims_master[20745]: Store Error: Unable to  
append cache for user/admin: File too large
```

You can determine this problem conclusively by looking at the file in the user’s message store directory, or by looking in the `imta` log file to see a more detailed message.

The immediate action is to reduce the size of the file. Either delete some mail, or move some of it to another mailbox. You could also use `mboxutil -r` to rename the folder out of the way, or `mboxutil -d` to delete the folder (see “[20.11.2.1 The mboxutil Utility](#)” on page [638](#)).

Long-term, you will need to inform the user of mailbox size limitations, implement an aging policy (see “[20.9 To Set the Automatic Message Removal \(Expire and Purge\) Feature](#)” on page 625), a quota policy (see “[20.8 About Message Store Quotas](#)” on page 616), set a mail box limit by setting `local.store.maxmessages`(see “[configutil Parameters](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*), set up an archiving system, or do something to keep the mailbox size under control.

20.14.4.9 IMAP Events Become Slow

Symptom: After working fine for a short period of time, many IMAP events become unreasonably slow, with some events taking over a second.

Diagnosis: You have the Event Notification Service (ENS) plugin, `libibiff`, configured, but ENS is not running or not reachable. See [Appendix B, “Administering Event Notification Service in Messaging Server,”](#) for ENS details.

Solution: If you want ENS notifications, make sure the ENS is enabled and configured correctly. If you do not want ENS notifications, make sure that `libibiff` is not being loaded. Typical bad configuration:

```
local.store.notifyplugin = /opt/sun/comms/messaging/lib/libibiff  
local.ens.enable = 0
```

Use either of the following for solution configurations:

```
local.store.notifyplugin =  
local.ens.enable = 0
```

or

```
local.store.notifyplugin = /opt/sun/comms/messaging/lib/libibiff  
local.ens.enable = 1
```

20.15 Migrating or Moving Mailboxes to a New System

It is sometimes necessary to move existing mailboxes from one Messaging Server system to another messaging server system. This commonly occurs in the following circumstances:

- Migrating from a non-Sun Messaging Server to the Sun Java System Messaging Server
- Moving mailboxes from one physical server to a different physical server

Messaging Server provides several ways to move mailboxes from one system to another. Each method has its advantages and disadvantages, which are described in the sections below. These methods are described in following sections:

- “[20.15.1 Migrating User Mailboxes to Another Messaging Server While Online](#)” on page 676

- “To Migrate User Mailboxes from One Messaging Server to Another While Online” on page 678
- “To Move Mailboxes Using an IMAP client” on page 683
- “To Move Mailboxes Using the MoveUser Command” on page 684
- “To Move Mailboxes Using the imsimport Command” on page 685

20.15.1 Migrating User Mailboxes to Another Messaging Server While Online

You can use this procedure to migrate the message store from an older version of Messaging Server to a newer version or to move mailboxes from one Sun Messaging Server message store to another. This procedure should work for iPlanet Messaging Server 5.0 and later. It cannot be used to move messages from earlier versions of Messaging Server or a non-Sun Microsystems message store.

The advantages of moving mailboxes using this procedure are as follows:

- System administrators move the mailboxes from the old source system to the new destination system without users involvement.
- This process is faster than any of the other processes.
- Re-linking is not required if you are moving an entire partition.
- Both Messaging Server systems remain active and online.
- You can migrate all the mailboxes on a messages store or a subset of those messages. This procedure allows for incremental migrations.

The disadvantages of moving mailboxes using this procedure are as follows:

- This method does not work with non-Sun messaging servers.
- The users being migrated will not have access to their mailboxes until the migration of their own mailbox is complete.
- This method can be complex and time consuming.

20.15.1.1 Incremental Mailbox Migration

Incremental migration provides numerous advantages for safely and effectively moving your message store to a different system or upgrading to a new system, incremental migration allows you to build a new back-end message store system alongside the old back-end message store. You can then test the new system, migrate a few friendly users, then test the new system again. Once you are comfortable with the new system and configuration, and you are comfortable with the migration procedure, you can start migrating real commercial users. These users can be split into discrete backup groups so that during migration, only members of this group are offline, and only for a short time.

Another advantage of on-line incremental migration is that you do not have to plan for a system-wide back out in case your upgrade fails. A back out is a procedure for reverting changes you have made to a system to return the system to the original working state. When doing a migration, you have to plan for failure, which means that for every step in the migration requires a plan to return your system back to its previous operational state.

The problem with offline migrations is that you can't be sure your migration is successful until you've completed all the migration steps and switched the service back on. If the system doesn't work and cannot be quickly fixed, you'll need a back out procedure for all the steps performed. This can be stressful and take some time, during which your users will remain offline.

With an on-line incremental migration you perform the following basic steps:

1. Build the new system alongside the old one so that both can operate independently.
2. Configure the old system for the coexistence with the new.
3. Migrate a group of friendly users and test the new system and its coexistence with the old system.
4. Divide the users on the old system into groups and migrate group by group to the new one as desired.
5. Disassemble the old system.

Because both systems will co-exist, you will have time to test and get comfortable with the new system before migrating to it. If you do have to perform a back out procedure--which should be very unlikely--you only have to plan for steps 2 and 4. Step 2 is easy to revert since you don't ever touch user's data. In step 4, the back out is to revert the user's state back to active and their mailhost attribute back to the old host. No system-wide back out is required.

20.15.1.2

On-line Migration Overview

Migrating mailboxes while remaining online is a straightforward process. Complications arise when you try to ensure that messages in transit to the mailbox (sitting in an MTA channel queue waiting for delivery) are not lost in the migration process. One solution is to hold messages sent during the migration process in a *held* state and wait for the messages in the various channel queues to be delivered. However, messages can get stuck in queues because of system problems or because a particular user is over quota. In this case, you must address this situation before migrating the mailboxes.

You can take various measures to reduce the likelihood of lost messages and to verify that messages are not stuck in a channel queue, but at a cost of increased complexity of the procedure.

The order and necessity of steps in the procedure vary depending upon your deployment and whether every message addressed to every mailbox must not be lost. This section describes the

theory and concepts behind the steps. It is incumbent on you to understand each step and decide which to take and in which order, given your specific deployment. Following is an overview of the process of moving mailboxes. This process might vary depending upon your deployment.

1. Block user access to the mailboxes being moved.
2. Temporarily hold messages addressed to the mailbox being moved.
3. Verify that messages are not stuck in the channel queues.
4. Change the user's mailhost attribute to the new mailbox location.
5. Move the mailboxes to the new location.
6. Release held mail to be delivered to the new mailbox and enable incoming messages to be delivered to the migrated mailboxes.
7. Examine the old message store to see if any messages were delivered after the migration.
8. Unblock user access to mailbox.

▼ To Migrate User Mailboxes from One Messaging Server to Another While Online

Before You Begin The requirements for this type of migration are as follows:

- stored should be running on both the source (old) and destination (new) messaging servers.
- The source system and destination system must be able to route messages to each other if both systems will operate in co-existence. This is needed, for instance, so that delivery status notification messages can be generated on the destination system and get delivered to the source system.

Note – Some steps apply only if you are upgrading the messaging server from an earlier version to a later version. These steps might not apply if you are only migrating mailboxes from one message store to another. The steps that apply to migrating entire systems are noted.

1 On the source system, split your user entries to be moved into equal backup groups using the backup-groups.conf file.

This step is in preparation for the mailbox migration, [Step 8](#), that occurs later in this procedure. See “[20.12.2 To Create Backup Groups](#)” on page [648](#) for detailed instructions.

You can also place the user names into files and use the -u option in the imsbackup command.

2 Notify users to be moved that they will not have access to their mailboxes until the move is completed.

Ensure that users to be moved are logged out of their mail systems before the data move occurs. (See “[20.13 Monitoring User Access](#)” on page 659.)

3 Set the authentication cache timeout to 0 on the back-end message store and MMP systems, and ALIAS_ENTRY_CACHE_TIMEOUT option to 0 on the MTAs.

a. On the back end message stores containing the mailboxes to be moved, set the authentication cache timeout to 0.

```
configutil -o service.authcachettl -v 0
```

This step and [Step 7](#) (changing `mailUserStatus` to `hold`) immediately prevents users from accessing their mailboxes during migration.

b. On all MMPs, set the LDAP and authentication cache timeout to 0.

In `ImapProxyAService.cfg` and `PopProxyAService.cfg` set both `LdapCacheTTL` and `AuthCacheTTL` to 0.

c. On any Messaging Server that hosts an MTA that inserts messages into mailboxes that are to be migrated, set the ALIAS_ENTRY_CACHE_TIMEOUT option to 0.

Messaging Servers hosting an MTA that inserts messages into the migrating mailboxes will typically be the back end message store. However, if the system is using LMTP, then that system will be the inbound MTAs. Check your configuration to make sure.

Resetting `ALIAS_ENTRY_CACHE_TIMEOUT` in `/msg_svr_base/config/option.dat` forces the MTA to bypass the cache and look directly at the LDAP entry so that intermediate channel queues (for example, the conversion or reprocess channels) see the new `mailUserStatus` (`hold`) of the users being moved rather than the out-of-date cached information.

`ALIAS_ENTRY_CACHE_TIMEOUT` is in `option.dat`.

d. Restart the systems on which the caches were reset.

You must restart the system for these changes to take place. See “[4.4 Starting and Stopping Services](#)” on page 124 for instructions.

4 Ensure that both your source Messaging Server and destination Messaging Server are up and running.

The source Messaging Server must be able to route incoming messages to the new destination server.

- 5 **Change the LDAP attribute `mailUserStatus` on all user entries whose mailboxes will be moved from `active` to `hold`.**

Changing the attribute holds incoming messages in the `hold` queue and prevents access to the mailboxes over IMAP, POP, and HTTP. Typically, users will be moved in groups of users. If you are moving all the mailboxes of a single domain, you can use the `mailDomainStatus` attribute.

For more information on `mailUserStatus`, see the “[mailUserStatus](#)” in *Sun Java Communications Suite 5 Schema Reference*.

- 6 **Make sure that messages addressed to mailboxes being migrated are not stuck in the `ims-ms` or `tcp_lmtp*` channel queues (if LMTP has been deployed).**

Use the following commands to see if messages exist in the channel queue directory tree and in the `held` state (to see `.HELD` files) addressed to a user to be migrated:

```
imsimta qm directory -to=<user_address_to_be_migrated> -directory_tree
```

```
imsimta qm directory -to=<user_address_to_be_migrated> -held -directory_tree
```

If there are messages in the queue, run these same commands later to see if the MTA has dequeued them. If there are messages that are not being dequeued, then you must address this problem before migrating. This should be a rare occurrence, but possible causes are recipient mailboxes being over quota, mailboxes being locked perhaps because users are logged in and moving messages, the LMTP backend server is not responding, network or name server problems, and so on).

- 7 **Change the LDAP attribute `mailHost` in the user entries to be moved as well as in any mail group entries*.**

Use the `ldapmodify` command to change the entries to the new mail server. Use the `ldapmodify` that comes with Messaging or Directory Server. Do NOT use the Solaris OS `ldapmodify` command.

* You only need to change the `mailHost` attribute in the mail group entry if the old mail host is being shut down. You can either change this attribute to the new mail host name or just eliminate the attribute altogether. It is optional for mail groups to have a `mailHost`. Having a `mailHost` means that only that host can do the group expansion; omitting a `mailHost` (which is the more common case) means all MTAs can do the group expansion. Note that mail group entries do not have mailboxes to be migrated and typically do not even have the `mailhost` attribute.

For more information on `mailhost`, see “[mailHost](#)” in *Sun Java Communications Suite 5 Schema Reference*.

8 Move the mailbox data from the source Messaging Server message store to the destination Messaging Server message store and record the time when started.

Back up the mailboxes with the `imsbackup` utility and restore them to the new Messaging Server with the `imsrestore` utility. For example, to migrate mailboxes from a Messaging Server 5.2 system called `oldmail.siroe.com` to `newmail.siroe.com`, run the following command on `oldmail.siroe.com`:

```
/server-root/bin/msg/store/bin/imsbackup -f- /instance/group      \
| rsh newmail.siroe.com /opt/SUNWmsgsr/lib/msg/imsrestore.sh      \
-f- -c y -v 1
```

You can run multiple concurrent backup and restore sessions (one per group) to maximize the transfer rate into the new message store. See the “[Command Descriptions](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for more information about the `imsbackup` and `imsrestore` utilities as well as “[20.12 Backing Up and Restoring the Message Store](#)” on [page 646](#).

Note – When `imsrestore` or any processing intensive operation takes significantly more system resources than normal, and continues doing so longer than the `msprobe` interval, there may be a temporary backlog of DB transaction log files to be cleared. If there are more files than specified in `local.store.maxlog`, then `msprobe` may erroneously restart all the processes during a restore. To prevent this from happening, disable `msprobe` during the `imsrestore`.

Note – Record the timestamp of when `imsbackup` is run for later delivery validation.

9 (Conditional Step for System Upgrades) If your mailbox migration is part of the process of upgrading from an earlier version of Messaging Server to the current version, set this current version of Messaging Server to be the new default Messaging Server for the system.

Change the DNS A record of `oldmail.siroe.com` to point to `newmail.siroe.com` (the server responsible for domain(s) previously hosted on `oldmail.siroe.com`).

10 Enable user access to the new message store.

Set the LDAP attribute `mailUserStatus` or `mailDomainStatus`, if applicable, to whatever value it had been before it was changed to `hold` (for example, `active`).

11 Release the messages in the `held` state on all source Messaging Servers.

Any system that may be holding incoming messages needs to run the following command to release all the user messages:

```
imsimta qm release -channel=hold -scope
```

where `scope` can be `all`, which releases all messages; `user`, which is the user ID; or `domain` which is the domain where the user resides.

12 Reset the authentication cache timeout and the ALIAS_ENTRY_CACHE_TIMEOUT option to the default or desired values and restart the system.

At this point, you've migrated all the user mailboxes that need to be migrated. Before proceeding, make sure that no new entries in LDAP have been created with the old system as the mailhost, and if some have, migrate them. Also, make sure that no such entries can be created by modifying the provisioning systems.

You will also want to change the `preferredmailhost` attribute to the name of the new mail host.

For back-end message stores, set authentication cache timeout as follows:

```
configutil -o service.authcachettl -v 900
```

For the MMPs, in `ImapProxyAService.cfg` and `PopProxyAService.cfg` set the `LdapCacheTTL` and `AuthCacheTTL` options to 900.

For MTAs, set the `ALIAS_ENTRY_CACHE_TIMEOUT` option to 600. `ALIAS_ENTRY_CACHE_TIMEOUT` is in `option.dat`.

You must restart the system for these changes to take place. See “[4.4 Starting and Stopping Services](#)” on page 124 for instructions.

13 Ensure that the user clients are pointing to the new mail server.

After the upgrade finishes, have the users point to the new server through their mail client program (in this example, users would point to `newmail.siroe.com` from `oldmail.siroe.com`).

An alternative is to use a messaging multiplexor (MMP) which obviates the need to have users point their clients directly to the new mail server. The MMP gets that information from the `mailHost` attribute which is stored in the LDAP user entries and automatically redirects the client to the new server.

14 After everything works, verify that no messages were delivered to the old message store after the migration.

Go to the old message store and run `mboxutil -l` to list the mailboxes. Check the last message delivery timestamp. If a message was delivered after the migration timestamp (the date stamp when you ran the `imsbackup` command), then migrate those messages with a backup and restore command. Because of the preparatory steps provided, it would be exceedingly rare to see a message delivered after migration.

Theoretically, a message could be stuck in a queue for the number of days or hours specified by the notices channel keywords (see “[10.10.4.3 To Set Notification Message Delivery Intervals](#)” on page 278).

15 Remove duplicate messages on the new message store, run the `relinker` command.

This command might free disk space on the new message store. See “[20.11.7 Reducing Message Store Size Due to Duplicate Storage of Identical Messages](#)” on page 643.

16 Remove the old messages from the store you migrated from and delete users from the database on the old store.

Run the `mboxutil -d` command. (See “[20.11.2.1 The mboxutil Utility](#)” on page 638).

▼ To Move Mailboxes Using an IMAP client

This procedure can be used anytime messages need to be migrated from one messaging server to a different messaging server. Consider the advantages and disadvantages before moving mailboxes using this method.

The advantages of moving mailboxes using IMAP clients are as follows:

- This method can be used to migrate from a non-Sun Messaging Server to the Sun Java System Messaging Server. It can also be used to move mailboxes from one physical server to a different physical server.
- After system administrators set up the new mail server or message store, responsibility for moving mailboxes to the new system is left to users.
- The process for moving mailboxes is relatively simple.
- User access to mailboxes does not have to be disabled.

The disadvantages of moving mailboxes using IMAP clients are as follows:

- Requires that both the old and new systems be simultaneously running and accessible to users.
- Cumulatively, this method takes longer to move mailboxes than other methods.
- Responsibility for moving mailboxes to the new system is left to users.
- The size of the new message store will be significantly larger than the old message store until the re-linking operation is performed.

1 Install and configure the new Messaging Server.

2 Set `local.store.relinker.enabled` to yes.

This will reduce the message store size on the new system caused by duplicate storage of identical messages. See “[20.11.7 Reducing Message Store Size Due to Duplicate Storage of Identical Messages](#)” on page 643 for more information.

3 Provision users on the new Messaging Server.

You can use Delegated Administrator to do this. As soon as users are provisioned on the new system, newly arriving mail will be delivered to the new INBOX.

- 4 **Have users configure their mail client to view both new and old Messaging Server mailboxes.**
This may involve setting up a new email account on the client. See mail client documentation for details.
- 5 **Instruct users to drag folders from their old Messaging Server to their new Messaging Server.**
- 6 **Verify with users that all mailboxes are migrated to the new system, then shut down the user account on the old system.**

▼ To Move Mailboxes Using the `MoveUser` Command

This procedure can be used anytime messages need to be migrated from one messaging server to a different messaging server. It is useful for migrating IMAP mailboxes from a non-Sun Messaging Server to the Sun Java System Messaging Server. Consider the advantages and disadvantages before moving mailboxes using this method.

The advantages of moving mailboxes using the `MoveUser` command are as follows:

- System administrators have complete responsibility for moving mailboxes from the old system to the new system. Users do not have to do anything.
- Works with any IMAP servers.

The disadvantages of moving mailboxes using the `MoveUser` command are as follows:

- Requires that both the old and new systems be simultaneously running and accessible to users.
- This method takes longer to move mailboxes than the other non-IMAP methods.
- Users access to mailboxes must be disabled while mailboxes are being moved.
- The size of the new message store will be significantly larger than the old message store until the re-linking operation is performed.

- 1 **Install and configure the new Messaging Server.**

- 2 **Set `local.store.relinker.enabled` to yes.**

This will reduce the message store size on the new system caused by duplicate storage of identical messages. See “[20.11.7 Reducing Message Store Size Due to Duplicate Storage of Identical Messages](#)” on page 643 for more information.

- 3 **Halt incoming mail to the messaging servers.**

Set the user attribute `mailUserStatus` to `hold`.

4 Provision users on the new Messaging Server if needed.

If you are migrating from a previous version of messaging server, you can use the same LDAP directory and server. MoveUser changes the mailhost attribute in each user entry.

5 Run the MoveUser command.

To move all users from host1 to host2, based on account information in the Directory Server siroe.com:

```
MoveUser -l \
"ldap://siroe.com:389/o=siroe.com???(mailhost=host1.domain.com)" \
-D "cn=Directory Manager" -w password -s host1 -x admin \
-p password -d host2 -a admin -v password
```

See the “[MoveUser](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for details on the MoveUser command.

6 Enable user access to the new messaging store.

Set the mailUserStatus LDAP attribute to active.

7 Shut down the old system.

▼ To Move Mailboxes Using the imsimport Command

This procedure is specifically used to move mailboxes from UNIX /var/mail format folders into a Sun Java System Messaging Server message store. However, if the messaging server from which you are migrating can convert the IMAP message stores to UNIX /var/mail format, then you can use the imsimport command to migrate messages to the Sun Java System Messaging Server. Consider the advantages and disadvantages before moving mailboxes using this method.

The advantage of moving mailboxes using the imsimport command is as follows:

- System administrators have complete responsibility for moving mailboxes from the old system to the new system. Users do not have to do anything.

The disadvantages of moving mailboxes using the imsimport command are as follows:

- This method takes longer to move mailboxes than the other non-IMAP methods.
- Users access to mailboxes must be disabled while mailboxes are being moved.
- The size of the new message store will be significantly larger than the old message store until the re-linking operation is performed.

1 Install and configure the new Messaging Server.

2 Set local.store.relinker.enabled to yes.

This will reduce the message store size on the new system caused by duplicate storage of identical messages. See “[20.11.7 Reducing Message Store Size Due to Duplicate Storage of Identical Messages](#)” on page 643 for more information.

3 Provision users on the new Messaging Server if needed.

You can use Delegated Administrator to do this. Do not switch over to the new system yet.

4 Disable user access to both the new and old messaging store.

Set the mailUserStatus LDAP attribute to hold. User’s mail is sent to the hold queue and access to the mailbox over IMAP, POP, and HTTP is disallowed. MTA and Message Access Servers on the store server must comply with this requirement. This setting overrides any other mailDeliveryOption settings.

5 If the mail store from the existing mail server is not already in the /var/mail format, convert the mail store to /var/mail files.

Refer to the third-party mail server documentation.

6 Run the imsimport command.

For example:

```
imsimport -s /var/mail/joe -d INBOX -u joe
```

See the “[imsimport](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for details on the imsimport command.

7 Enable user access to the message store.

Set the mailUserStatus LDAP attribute to active.

8 Enable user access to the new messaging store.**9 Shut down the old system.**

Message Archiving

This chapter describes archiving concepts for Messaging Server. It does not provide instructions on how to set up an archiving system. Refer to [Message Archiving Using the Sun Compliance and Content Management Solution](#) for more detailed deployment instructions. The reason for putting the deployment information in a separate document is so that configuration updates can be made more quickly.

This chapter consists of the following sections:

- “[21.1 Archiving Overview](#)” on page 687

21.1 Archiving Overview

A message archiving system saves all or specified incoming and outgoing messages on a system separate from Messaging Server. Sent, received, deleted, and moved messages can all be saved and retrieve in an archive system. Archived messages cannot be modified or removed by email users, so the integrity of incoming and outgoing messages is maintained. Message archiving is useful for compliance record keeping, but it is also useful for message store management. For example, some customers may use archiving to perform message back-up or to move older messages from more expensive message store storage to less expensive archive storage.

Archived messages can be accessed through a separate archiving software GUI client or through Messaging Server. If the messages are deleted from Messaging Server, then the archiving client can be used to search for and retrieve those deleted messages since archived messages are never deleted. Note, however, that archived messages are not stored in mailbox folders as they are in the Messaging Server.

The system can also be set up so that archived messages can be accessed from Messaging Server. For example, you can set up a system to archive messages over 2 years old. Instead of having message bodies reside in the message store, they would instead reside in the archive system. From the users standpoint, the message appears no different from a regular email message. The same header and subject information will appear (this is still stored in the message store).

storage), but the message body is downloaded from the archive server by the message store when needed. Thus, there may be a slight delay as messages are downloaded from the archive server. In addition, archived messages cannot be searched from the email client. Searching must be done from the archiving GUI.

21.1.1 Message Archiving Systems: Compliance and Operational

There are two types of archiving, compliance and operational. Compliance archiving is used when you have a legal obligation to maintain strict retrievable email record keeping. Selected email (selected by user(s), domain, channel, incoming, outgoing and so on) coming into the MTA is copied to the archive system before being delivered to the message store or the internet. Archiving can be set to occur either before or after spam and virus filtering.

Operational archiving is used for mail management purposes. For example:

- To reduce storage usage on the Messaging Server message store by moving less used (older) messages to an archiving system which uses lower cost storage.
- As an alternative for data backup.

Note that compliance and operational archiving are not exclusive. That is, you can set up your system so that it does both compliance and operational archiving.

Configuring the JMQ Notification Plug-in to Produce Messages for Message Queue

This chapter describes how to configure a JMQ notification plug-in to produce messages to be consumed by clients in a Message Queue service.

This chapter contains the following sections:[Chapter 22, “Configuring the JMQ Notification Plug-in to Produce Messages for Message Queue”](#)

- “[22.1 JMQ Notification Overview](#)” on page 689
- “[22.2 Configuring a JMQ Notification Service](#)” on page 692
- “[22.3 JMQ Notification Messages and Properties](#)” on page 701

22.1 JMQ Notification Overview

The Messaging Server notification plug-in allows you to deliver notification messages to a messaging service or event service. The messaging service sends the notifications to consumers (client interfaces), which filter and deliver the messages to specified users.

For example, when new email arrives in a user's mailbox, the notification plug-in delivers a notification message to the messaging service. The message consumer, a component of the messaging service, receives the notification and sends it to the user's email client (such as Communications Express or Mozilla mail). The email client can then display a pop-up on the user's computer screen: “You have received a new message.”

Another example: if a user's mailbox exceeds its quota, the notification plug-in produces an over-quota notification message. The message consumer sends a warning to the user and to an administrator who needs to be informed of the event.

22.1.1 Two Notification Messaging Services

You can configure Messaging Server to deliver notifications to two different messaging services:

- Sun Java System Message Queue 3.6 2005Q4

- Event Notification Service

The Message Queue service implements the Java Messaging Service (JMS) specification, providing a message broker, interfaces to create clients that produce or consume messages, and administrative services and control. Message Queue follows the JMS standard for routing and delivery functions, protocols, and message formats.

The Event Notification Service is a component bundled with Messaging Server and Sun Java System Calendar Server. It is a proprietary service that uses a publish/subscribe architecture for sending and receiving event notifications.

You can configure a notification producer for Message Queue, for the Event Notification Service, or for both services.

Note – This chapter describes how to configure notifications for Message Queue only.

For information about the Event Notification Service, see the *Sun Java System Communications Suite Event Notification Service Guide*.

22.1.2 Notification Plug-ins

To enable Messaging Server to produce notifications to Message Queue or the Event Notification Service, you must configure a plug-in for that service:

- The JMQ notification plug-in allows you to deliver notification messages to the Message Queue broker.
- The iBiff plug-in allows you to publish notification events to the Event Notification Service.

For information on how to load the iBiff plug-in and configure the Event Notification Service, see “Appendix B: Administering Event Notification Service in Messaging Server,” in the *Sun Java System Messaging Server Administration Guide*.

22.1.3 Benefits of Using JMQ Notification

The JMQ notification plug-in, with Message Queue, provides the following benefits:

- Message Queue implements the JMS standard.
- With Message Queue, you can produce messages to a *topic* or a *queue*, or to both of these delivery methods. For a brief definition, see “[22.1.3.1 Publishing to a Topic or a Queue](#)” on [page 691](#).
- Message Queue offers enhanced load balancing during message distribution, especially when messages are produced to a queue.

- The JMQ notification plug-in allows you to configure up to five notification plug-ins. The different plug-ins can produce messages to a topic, to a queue, to the Event Notification Service, and so on. For details, see “[22.1.3.2 Using Multiple JMQ Notification Plug-ins](#)” on page 691.
- Message Queue provides reliable delivery of notifications.

For example, if you configure the JMQ notification plug-in to produce messages with the persistent flag enabled, the message remains in the Message Queue broker until a consumer receives it. The message is saved so that, if a server goes down, the message can be retrieved and made available to the appropriate consumer.

22.1.3.1

Publishing to a Topic or a Queue

A topic and queue use different messaging delivery patterns; each one can be configured in a Message Queue service.

Topic. When a message producer sends a message to a topic, a publish/subscribe architecture is used. In this broadcast pattern, a producer sends a message to a topic destination. Any number of consumers can be subscribed to this topic destination. Each consumer subscribed to the topic gets its own copy of the message. If no consumers are subscribed to the topic, the message is discarded.

The Event Notification Service also uses a publish/subscribe architecture; it is similar to the topic pattern defined in Message Queue.

Queue. When a message producer sends a message to a queue, a point-to-point architecture is used. In this pattern, a producer sends a message to a queue destination from which only one consumer can receive it. If several consumers are waiting for messages from the queue, only one subscriber will receive the message. If no consumers are waiting, the message is held until either the message times out, or a consumer expresses an interest in the queue.

Producing messages to a queue allows you to spread the message load across multiple consumers.

22.1.3.2

Using Multiple JMQ Notification Plug-ins

You can configure from one to five notification plug-ins.

Messaging Server provides a plug-in library at the following default location:

`/opt/SUNWmsgsr/lib/libjmqnotify`

You use the `configutil` utility to specify parameters for a plug-in and to point the plug-in to the library of executable code.

If you specify more than one plug-in, each plug-in produces notification messages independently of the others. For example, if two plug-ins are configured with a delete-message parameter, and a message is deleted from a user's mailbox, both plug-ins will produce a notification message.

By configuring multiple plug-ins, you can take advantage of different message-distribution patterns for different purposes. For example, you could configure three different plug-ins to produce messages

- To a queue (using Message Queue)
- To a topic (using Message Queue)
- To the Event Notification Service

22.1.3.3

Configuring Parameters for a Notification Plug-in

For each plug-in you configure, you must define a separate set of configutil parameters.

The parameters determine two kinds of information:

- The kinds of notification messages to produce. For example, enabling the LogUser parameter causes a notification message to be sent whenever a user logs in or out.
- Configuration information needed by Message Queue. For example, the jmqHost parameter identifies the IP address of the host where the Message Queue broker is running.

For instructions on how to configure a plug-in, see “[To Configure a JMQ Notification Plug-in](#)” on page 693.

22.2 Configuring a JMQ Notification Service

This section briefly describes how a JMQ notification plug-in fits into the context of a complete Message Queue service. It then provides detailed instructions for configuring a JMQ notification plug-in.

22.2.1

Planning for Your JMQ Notification Service

A JMQ notification plug-in is only one part of a Message Queue service. A messaging service also includes clients that consume the messages and the Message Queue infrastructure (the broker, administration components, and so on).

The following steps outline the tasks you should perform to create a Message Queue service that supports Messaging Server:

1. Design your notification message service.

Define the notification messages needed for your Messaging Server installation. The planning and design phases of your message-service development lifecycle lie outside the scope of this chapter. However, you should answer the following design questions before you configure the JMQ notification plug-in:

- Which message events do you need to produce notifications? For a list of the available notification messages, see “[22.3.1 Notification Messages](#)” on page 701.

- Do you intend to produce messages to a queue, a topic, or both?
- Do you intend to use the proprietary Event Notification Service as well as the Message Queue service?

The answers to these questions will help you decide whether to configure one notification plug-in or multiple plug-ins, and to determine how to configure each plug-in.

2. Install, configure, and deploy the Message Queue product.

For information about installing Message Queue, see the *Sun Java System Message Queue Installation Guide*.

For information about configuring and deploying Message Queue, see the *Sun Java System Message Queue Administration Guide*.

3. Write one or more Message Queue clients that will consume the JMQ notification messages.

The clients must conform to the requirements for a Message Queue client API. A simple example of client source code, written in C, is available in the following path:

```
/opt/SUNWmsgsr/examples/jmq sdk/
```

The source file name is `jmqclient.c`.

This client source code receives messages from the JMQ notification messages defined by the parameters in the `libjmqnotify` library. It then sends the message output to `stdout`.

For information about writing Message Queue clients in C or Java, see the *Sun Java System Message Queue Developer's Guide for C Clients* or the *Sun Java System Message Queue Developer's Guide for Java Clients*.

4. Configure and enable the JMQ notification plug-in for producing notification messages.

The remainder of this chapter describes how to configure the notification plug-in.

5. Configure and start the runtime Message Queue clients.

For information about deploying the runtime Message Queue clients, see the *Sun Java System Message Queue Administration Guide*.

▼ To Configure a JMQ Notification Plug-in

In this procedure, you first configure the message events that will produce notifications. Next, you specify the information needed by Message Queue. Finally (in step 9), you configure the plug-in name by specifying a parameter after the name of the plug-in library:

```
'/opt/SUNWmsgsr/lib/libjmqnotify$plug-in_name'
```

If you do not specify a plug-in name, `jmqnotify` is used by default.

Before You Begin You should install, configure, and deploy the following products:

- Sun Java System Messaging Server
 - Sun Java System Message Queue 3.6 SP3 2005Q4 or later
-

Note – Most of the `configutil` parameters you will configure in the following steps are optional. For a list of their default values, see [Table 22–2](#).

1 Configure the notification message parameters.

For each kind of notification message you want to include in the plug-in, use the `local.store.notifyplugin` command with the `configutil` utility.

For example, to enable notifications for new messages, enter:

```
configutil -o local.store.notifyplugin.jmqnotify.NewMsg.enable -v 1
```

where `jmqnotify` is the name of the plug-in

and `-v 1` enables notifications for this message. A value of `0` disables notifications for this message.

For a list of all the JMQ notification messages, see [“22.3.1 Notification Messages” on page 701](#).

For definitions of the `configutil` parameters that enable the JMQ notification messages, see “Chapter 3: Messaging Server Configuration,” in the *Sun Java System Messaging Server Administration Reference*.

A few notification messages use more than one `configutil` parameter to enable the message with additional features. For example, some messages can carry message headers in the notification text. For instructions on how to configure these messages, see [“Syntax for newflags and oldflags Properties” on page 710](#).

Note – You must configure parameters separately for each plug-in you configure.

Thus, if you configure two plug-ins, named `jmq1` and `jmq2`, and you want to enable new-message notifications for both plug-ins, you must run the `local.store.notifyplugin` command twice:

```
configutil -o local.store.notifyplugin.jmq1.NewMsg.enable -v 1
```

```
configutil -o local.store.notifyplugin.jmq2.NewMsg.enable -v 1
```

2 Specify the host where the Message Queue destination (broker) is running.

For example, enter the following command:

```
configutil -o local.store.notifyplugin.jmqnotify.jmqHost -v "127.0.0.1"
```

where `jmqnotify` is the name of the plug-in

and "127.0.0.1" is the IP address of the host machine for the Message Queue broker.

3 Specify the port for the Message Queue broker.

For example, enter the following command:

```
configutil -o local.store.notifyplugin.jmqnotify.jmqPort -v "7676"
```

where *jmqnotify* is the name of the plug-in

and "7676" is the port for the Message Queue broker.

4 Specify the user ID and password of the Message Queue user authorized to produce messages to the service.

For example, enter the following commands:

```
configutil -o local.store.notifyplugin.jmqnotify.jmqUser -v "guest"
```

```
configutil -o local.store.notifyplugin.jmqnotify.jmqPwd -v "%$#a62t&"
```

where *jmqnotify* is the name of the plug-in

and "guest" and "%\$#a62t&" are the user ID and password, respectively, for the Message Queue user.

5 Configure the type of destination (topic or queue) and the name of the destination to which messages will be sent.

Follow these steps:

a. Specify whether the destination is a topic or queue.

For example, enter the following command:

```
configutil -o local.store.notifyplugin.jmqnotify.DestinationType -v "queue"
```

where *jmqnotify* is the name of the plug-in

and "queue" specifies that the destination is a queue. The allowed values for this parameter are "queue" and "topic".

b. Specify the destination name.

For example, enter one of the following commands:

```
configutil -o local.store.notifyplugin.jmqnotify.jmqQueue -v "JES-MS"
```

or

```
configutil -o local.store.notifyplugin.jmqnotify.jmqTopic -v "JES-MS"
```

where *jmqnotify* is the name of the plug-in

`jmqQueue` or `jmqTopic` identifies the destination type. The `jmqQueue` and `jmqTopic` parameters are synonymous and mutually exclusive; you can only use one of these parameters in one plug-in.

"JES-MS" is an example name of the queue or topic to which messages will be sent.

6 Specify the message priority.

For example, enter the following command:

```
configutil -o local.store.notifyplugin.jmqnotify.Priority -v 3
```

where `jmqnotify` is the name of the plug-in

and `-v 3` is the Message Queue priority assigned to messages produced by this plug-in.

The default value of the Priority is 4.

7 Specify the length of time (in milliseconds) that messages are retained by the Message Queue broker.

For example, enter the following command:

```
configutil -o local.store.notifyplugin.jmqnotify.ttl -v 100
```

where `jmqnotify` is the name of the plug-in

and `-v 100` specifies that a message is retained by the Message Queue service for 100 milliseconds before being either delivered or discarded. A value of 0 means that a message is retained permanently; it does not time out.

8 Specify the message persistence.

For example, enter the following command:

```
configutil -o local.store.notifyplugin.jmqnotify.Persistent -v 1
```

where `jmqnotify` is the name of the plug-in

and `-v 1` specifies that persistent messages are used in the Message Queue service. Allowed values are 1 (persistent) and 0 (non-persistent).

9 Configure the plug-in name.

To configure a single plug-in with the default name, you can enter either the fully qualified name of the plug-in library or the name of the library and its plug-in parameter:

```
configutil -o local.store.notifyplugin -v /opt/SUNWmsgsr/lib/libjmqnotify
```

or

```
configutil -o local.store.notifyplugin -v '/opt/SUNWmsgsr/lib/libjmqnotify$jmqnotify'
```

where `libjmqnotify` is the library name

and `jmqnotify` is the default name of the plug-in parameter.

Use the dollar sign (\$) to separate the library name from the parameter.

Enclose the entire value in single quotes ('*value*'); if you do not, the shell will interpret the dollar sign.

The configutil parameters read by the default plug-in have the following names:

```
local.store.notifyplugin.jmqnotify.*
```

To configure a different plug-in name such as jmq42, you would enter the following command:

```
configutil -o local.store.notifyplugin -v '/opt/SUNWmsgsr/lib/libjmqnotify$jmq42'
```

The configutil parameters read by the jmq42 plug-in have the following names:

```
local.store.notifyplugin.jmq42.*
```

▼ To Configure Multiple Plug-ins

1 Configure a separate set of JMQ notification parameters for each plug-in you intend to create.

For example, suppose you configure two plug-ins named jmq1 and jmq2. Assume you want to enable new-message notifications for both plug-ins and purged-message notifications for the jmq2 plug-in only. In this case, you would run the local.store.notifyplugin command three times, as follows:

```
configutil -o local.store.notifyplugin.jmq1.NewMsg.enable -v 1
```

```
configutil -o local.store.notifyplugin.jmq2.NewMsg.enable -v 1
```

```
configutil -o local.store.notifyplugin.jmq2.PurgeMsg.enable -v 1
```

You also must specify parameters that enable the plug-ins to communicate with the Message Queue service.

For step-by-step instructions for configuring the all the notification parameters, see “[To Configure a JMQ Notification Plug-in](#)” on page 693.

2 Configure the plug-in names.

To configure two plug-ins named jmq1 and jmq2, you would enter the following command:

```
configutil -o local.store.notifyplugin
-v '/opt/SUNWmsgsr/lib/libjmqnotify$jmq1$$/opt/SUNWmsgsr/
lib/libjmqnotify$jmq2'
```

In this example, two instances of the plug-in library are run.

Use a single dollar sign (\$) to separate the library name from the parameter that specifies the plug-in name.

Use two dollar signs (\$\$) to separate the first plug-in instance from the next one.

Enclose the entire value in single quotes ('*value*'); if you do not, the shell will interpret the dollar signs.

In this example, the first instance builds its configuration from parameters with the name `jmq1`:

```
local.store.notify.jmq1.*
```

The second instance builds its configuration from parameters with the name `jmq2`:

```
local.store.notify.jmq2.*
```

22.2.2 Specifying Notification Messages that Use More Than One configutil Parameter

For most notification messages, you specify the message by running a single `local.store.notifyplugin` command.

However, the following notification messages are (or can be) configured with more than one `local.store.notifyplugin` command:

1. NewMsg
2. UpdateMsg
3. DeleteMsg
4. MsgFlags

The procedures that follow describe how to set up these notification messages.

▼ To Configure New-Message and Updated-Message Notifications with Message Headers and Message Bodies

You can add the message headers and message bodies to the text of notification messages sent when there are new or updated email messages.

Including message headers and message bodies is optional; you can include both features, one feature only, or neither feature. The default is to send messages without message headers or message bodies.

1 Specify the new-message or updated-message notification:

```
configutil -o local.store.notifyplugin.jmqnotify.NewMsg.enable -v 1
```

```
configutil -o local.store.notifyplugin.jmqnotify.UpdateMsg.enable -v 1
```

where `jmqnotify` is the name of the plug-in

and `-v 1` enables notifications for these messages. A value of `0` disables notifications.

- 2 Specify the `maxHeaderSize` parameter with a value greater than zero, as in the following example:

```
configutil -o local.store.notifyplugin.jmqnotify.maxHeaderSize -v 1024
```

where *jmqnotify* is the name of the plug-in

and 1024 is the maximum size of the header to be sent. The default value of `maxHeaderSize` is 0, which sends no header information with the message.

- 3 Specify the `maxBodySize` parameter with a value greater than zero, as in the following example:

```
configutil -o local.store.notifyplugin.jmqnotify.maxBodySize -v 1024
```

where *jmqnotify* is the name of the plug-in

and 1024 is the maximum size of the message body to be sent. The default value of `maxBodySize` is 0, which sends no body with the message.

▼ To Configure Deleted-Message Notifications with Message Headers

You can add the message headers to the text of notification messages sent when email messages are deleted.

Including message headers is optional. The default is to send notifications without message headers.

- 1 Enable notifications to be sent when email messages are deleted:

```
configutil -o local.store.notifyplugin.jmqnotify.DeleteMsg.enable -v 1
```

where *jmqnotify* is the name of the plug-in

and -v 1 enables notifications for this message. A value of 0 disables notifications.

- 2 Specify the `ExpungeHeaders` parameter:

```
configutil -o local.store.notifyplugin.jmqnotify.ExpungeHeaders -v 1
```

where *jmqnotify* is the name of the plug-in

and -v 1 enables message headers to be carried with deleted-message notifications. The default value of `ExpungeHeaders` is 0, which prohibits deleted-message notifications from carrying header information.

You must configure the `ExpungeHeaders` parameter to enable `DeleteMsg` messages to carry message headers.

- 3 Specify the `maxHeaderSize` parameter with a value greater than zero, as in the following example:

```
configutil -o local.store.notifyplugin.jmqnotify.maxHeaderSize -v 1024
```

where *jmqnotify* is the name of the plug-in

and 1024 is the maximum size of the header to be sent. The default value of `maxHeaderSize` is 0, which sends no header information with the message.

22.2.2.1

Configuring Notifications for Changes in Message Status

You can configure a notification message to be sent when an email message has changed status.

Information Delivered in Message-Flag Notifications

A message-flags notification is produced whenever a status flag has changed because the email message was:

- Answered
- Flagged
- Deleted
- Seen (read)
- Draft

When a message-flags notification is sent, the notification carries the following properties:

- The flags set for the email message before its status changed
- The flags set for the email message after its status changed

This information is carried in two properties, `oldflags` and `newflags`, which are 5-character strings.

For a description of the values of these two properties, see “[Syntax for newflags and oldflags Properties](#)” on page 710.

Configutil Parameters Needed for Message-Flag Notifications

To enable message-flag notifications, you must configure the following `configutil` parameters:

- `local.store.notifyplugin.MsgFlags`
- `local.store.notifyplugin.*.MsgFlags.enable`

The first `MsgFlags` parameter enables the IMAP server and message store to identify and track the changing values of the status flags so that this information can be delivered in notification messages.

This parameter applies to all notification plug-ins. Therefore, you must enable the parameter if *any* notification plug-in uses message-flag notifications. If no plug-in uses message-flag notifications, be sure that this parameter is disabled (its default value).

The second parameter, `*.MsgFlags.enable`, allows message-flag notifications to be sent for a particular plug-in library.

Note – You must configure both parameters to enable notifications for message flags.

▼ To Enable Notifications When Message-Status Flags Have Changed

- 1 **Enable status flags to be tracked and status information to be carried with message-flag notifications:**

```
configutil -o local.store.notifyplugin.MsgFlags -v 1
```

where `-v 1` enables message-flag information to be sent with message-flag notifications. A value of `0` disables this notification.

- 2 **Enable message-flag notifications to be sent by a particular plug-in:**

```
configutil -o local.store.notifyplugin.jmqnotify.MsgFlags.enable -v 1
```

where `jmqnotify` is the name of the plug-in

and `-v 1` enables message-flag notifications for this plug-in. A value of `0` disables notifications.

22.3 JMQ Notification Messages and Properties

This section describes the following topics:

- “22.3.1 Notification Messages” on page 701
- “22.3.2 Rules and Guidelines for Notification Messages” on page 703
- “22.3.3 Notifications for Particular Message Types” on page 703
- “22.3.4 Default Values of the configutil Parameters” on page 704
- “22.3.5 Notification Message Properties” on page 705

22.3.1 Notification Messages

Notification messages can be generated for various kinds of events that occur in the message store. For example, when a user logs in, a `Login` message can be produced and delivered to the Message Queue broker.

A `configutil` parameter specifies each kind of message to be produced. You determine which events will generate messages by configuring various `configutil` parameters. The `configutil` parameters are referenced by one or more JMQ Notification plug-in libraries.

All messages are delivered to a topic or a queue, depending on whether the destination type is set to “topic” or “queue”. For information on how to configure the Message Queue destination, see “[To Configure a JMQ Notification Plug-in](#)” on page 693.

Each message is identified by the following message header:

MQ_MESSAGE_TYPE_HEADER_PROPERTY

The JMQ Notification plug-in supports the messages shown in the following table.

For a list of the `configutil` parameters that enable these messages, see “[22.3.4 Default Values of the configutil Parameters](#)” on page [704](#).

TABLE 22-1 JMQ Notification Messages

Notification Message	Description
DeleteMsg	Messages marked as “Deleted” are removed from the mailbox. This is the equivalent to IMAP expunge.
Login	User logged in from IMAP, HTTP, or POP. (This message is enabled with the <code>configutil</code> parameter <code>local.store.notifyplugin.*.LogUser.enable</code> .)
Logout	User logged out from IMAP, HTTP, or POP. (This message is enabled with the <code>configutil</code> parameter <code>local.store.notifyplugin.*.LogUser.enable</code> .)
MsgFlags	Message flags on a message have been changed. The old and new flags are carried with this message.
NewMsg	New message was received by the system into the user’s mailbox. Can contain message headers and body.
OverQuota	Operation failed because the user’s mailbox exceeded one of the quotas (diskquota, msgquota). The MTA channel holds the message until the quota changes or the user’s mailbox count goes below the quota. If the message expires while it is being held by the MTA, it will be expunged.
PurgeMsg	Message expunged (as a result of an expired date) from the mailbox by the server process imexpire. This is a server side expunge, whereas DeleteMsg is a client side expunge. This is not a purge in the true sense of the word.
ReadMsg	Message in the mailbox was read. (In the IMAP protocol, the message was marked Seen.)
TrashMsg	Message was marked for deletion by IMAP or HTTP. The user may still see the message in the folder, depending on the mail client’s configuration. The messages are to be removed from the folder when an expunge is performed.
UnderQuota	Quota went back to normal from OverQuota state.
UpdateMsg	Message was appended to the mailbox by an IMAP operation. For example, the user copied an email message to the mailbox. Can contain message headers and body.

22.3.2 Rules and Guidelines for Notification Messages

The following rules and guidelines apply to the supported notification messages:

- The text of most notification messages is a single blank space. (The blank space is used because Message Queue does not permit an empty message body.) The exceptions are as follows:
 - The NewMsg, UpdateMsg, and DeleteMsg messages can include a message header when configured with the `maxHeaderSize` parameter. You must set `maxHeaderSize` to a value greater than zero.
To include a message header with a DeleteMsg message, you also must set the `ExpungeHeaders` parameter to a value of 1.
 - NewMsg and UpdateMsg message can include a message body when configured with the `maxBodySize` parameter. You must set `maxBodySize` to a value greater than zero.
For NewMsg and UpdateMsg, by default the message body is not delivered (is turned off). This prevents overloading Message Queue. No other messages include a message body.
 - Notification messages can be generated for changes to the INBOX alone, or to the INBOX and all other folders. The following configuration parameter allows for INBOX only (value = 0), or for both the INBOX and all other folders (value = 1):

```
local.store.notifyplugin.jmqnotify.noneInbox.enable
```

The default setting is to generate messages from the INBOX only (value = 0).

There is no mechanism to select folders; all folders are included when the variable is enabled (value = 1).

- The NewMsg notification is issued only after the message is deposited in the user mailbox (as opposed to “after it was accepted by the server and queued in the message queue”).
- Messages are not generated for POP3 client access.
- All messages can be suppressed by issuing XNOTNOTIFY. For example, an IMAP script used for housekeeping only (the users are not meant to be notified) might issue it to suppress all messages.

22.3.3 Notifications for Particular Message Types

Notifications can deliver status information about messages of different types, such as text messages, voice mail, and image data. Users often expect these heterogeneous message types to be stored in the same mail folder. For example, a user may want new text messages and voice mail to arrive in the user's cell phone inbox.

To configure these message types, you use configutil commands such as `store.messagetype.enable`. For information about configuring and managing message types, see “Managing Message Types” in “Chapter 18: Managing the Message Store.”

Once the message types have been configured, JMQ notification messages can identify the particular message types. You can write your Message Queue client to interpret notification messages by message type and deliver status information about each type to the mail client.

For example, suppose new messages of different types arrive in a user's mailbox. A NewMsg notification message can carry data to tell the user that, for example, there are seven new voice mail messages and four new text messages in the user's inbox.

The following notification messages can carry information that tracks particular message types:

```
NewMsg  
UpdateMsg  
ReadMsg  
TrashMsg  
DeleteMsg  
PurgeMsg  
OverQuota  
UnderQuota
```

The JMQ notification function counts the number of messages currently in the mailbox, by message type. Instead of sending one count, an array specifying the count for each message type is sent with the notification message.

The message-specific count is carried in the numMsgs property and delivered with the notification message. For ReadMsg and TrashMsg notification messages, the number of messages seen (numSeen) and the number marked as deleted (numDeleted) are also counted by message type.

Note – The Event Notification Service does not support message types. Use a JMQ notification plug-in to deliver information about message types.

22.3.4

Default Values of the configutil Parameters

The notification messages and the configuration information needed by Message Queue are configured with configutil parameters.

Table 22–2 shows these parameters and their default values.

For complete definitions of the configutil parameters, see “Chapter 3: Messaging Server Configuration,” in the *Sun Java System Messaging Server Administration Reference*.

TABLE 22–2 configutil Parameters and Their Default Values

configutil Parameter	Default Value
local.store.notifyplugin.*.maxBodySize	0 — Disabled
local.store.notifyplugin.*.maxHeaderSize	0 — Disabled
local.store.notifyplugin.*.NewMsg.enable	1 — Enabled
local.store.notifyplugin.*.UpdateMsg.enable	1 — Enabled
local.store.notifyplugin.*.ReadMsg.enable	1 — Enabled
local.store.notifyplugin.*.DeleteMsg.enable	1 — Enabled
local.store.notifyplugin.*.PurgeMsg.enable	1 — Enabled
local.store.notifyplugin.*.LogUser.enable	1 — Enabled
local.store.notifyplugin.*.MsgFlags.enable	0 — Disabled
local.store.notifyplugin.*.noneInBox.enable	0 — Disabled
local.store.notifyplugin.*.jmqHost	“127.0.0.1”
local.store.notifyplugin.*.jmqPort	7676
local.store.notifyplugin.*.jmqTopic	“JES-MS”
local.store.notifyplugin.*.jmqQueue	“JES-MS”
local.store.notifyplugin.*.jmqUser	“guest”
local.store.notifyplugin.*.jmqPwd	“guest”
local.store.notifyplugin.*.destinationtype	“topic”
local.store.notifyplugin.*.Priority	4
local.store.notifyplugin.*.ttl	0 — Indicates that messages never time out.
local.store.notifyplugin.*.Persistent	1 — Enabled

22.3.5

Notification Message Properties

Every message carries additional information defined in properties. Different properties are present for different messages. For example, NewMsg indicates the IMAP uid of the new message.

22.3.5.1

Standard Notification Message Properties

Table 22–3 describes the standard notification message properties. These properties are present in all JMS messages.

TABLE 22–3 Standard Notification Message Properties

Property	Data Type	Description
hostname	ConstMQString	The host name of the machine that generated the message.
pid	MQInt32	ID of the process that generated the message.
process	ConstMQString	Specifies the name of the process that generated the message.
timestamp	MQFloat64	Specifies the number of milliseconds since the epoch (midnight GMT, January 1, 1970).

22.3.5.2

Properties Specific to Particular Notification Messages

[Table 22–4](#) describes the properties carried with particular notification messages.

Each message includes a subset of the properties shown in the table below. For a list of the properties associated with each message, see [Table 22–5](#).

TABLE 22–4 Properties Specific to Particular Notification Messages

Property	Data Type	Description
client	ConstMQString	The IP address of the Message Queue client associated with the message.
diskquota	MQInt32	The disk space quota, in kilobytes, for the user associated with the message. The value is set to -1 to indicate no quotas.
diskquotused	MQInt32	The amount of disk space used by the user associated with the message, in kilobytes.
hdrLen	MQInt32	The size of the message header. Note that this might not be the size of the header in the message body, because it might have been truncated.
imapUid	MQInt32	The IMAP uid property associated with the message.
lastUid	MQInt32	The last IMAP uid value used in the mailbox.

TABLE 22–4 Properties Specific to Particular Notification Messages (Continued)

Property	Data Type	Description
mailboxName	ConstMQstring	The message-store mailbox name associated with the event. The mailboxName has one of the following formats (where uid is the user's unique identifier): uid — identifies the inbox of a user in the default (primary) domain. uid@domain — identifies the inbox of a user in a hosted domain. uid/mailboxname — identifies the top-level mailbox of a user in the default domain. uid@domain/mailboxname — identifies the top-level mailbox of a user in a hosted domain. uid/foldername/mailboxname — identifies a mailbox in a folder of a user in the default domain. uid@domain/foldername/mailboxname — identifies a mailbox in a folder of a user in a hosted domain.
msgquota	MQInt32	The user's quota for the maximum number of messages. The value is set to -1 to indicate no quotas.
newflags	ConstMQString	The flags set for the user's mailbox message after they were changed by the current operation. This property is always present, together with oldflags, when a <code>MsgFlags</code> notification message is produced. For the syntax and values for newflags, see “ Syntax for newflags and oldflags Properties ” on page 710, below this table.
numDeleted	MQInt32	The number of messages in the mailbox marked as deleted. This number counts the messages deleted by the mailbox owner. If other users have access to the mailbox, their actions in the mailbox are not included in this count. (However, the other users' actions can trigger notifications such as <code>DeleteMsg</code>).

TABLE 22-4 Properties Specific to Particular Notification Messages *(Continued)*

Property	Data Type	Description
numDeleted <i>nn</i>	MQInt32	<p>The total number of messages in the mailbox marked as deleted, specified for each message type. If message types are configured, a numDeleted<i>nn</i> property carries a count for each message type <i>nn</i>.</p> <p>The numDeleted property is always sent; it counts the total number of all messages marked as deleted, including all types.</p> <p>For example, if 20 messages are marked as deleted, 10 are of type 3, 7 are of type 16, and the rest are not of any recognized type, the following properties and counts are carried with the notification:</p> <pre>numDeleted=20 numDeleted3=10 numDeleted16=7</pre>
numMsgs	MQInt32	<p>The total number of messages now in the mailbox.</p>
numMsgs <i>nn</i>	MQInt32	<p>The total number of messages now in the mailbox, specified for each message type. If message types are configured, a numMsgs<i>nn</i> property carries a count for each message type <i>nn</i>.</p> <p>The numMsgs property is always sent; it counts the total number of all messages in the mailbox, including all types.</p> <p>For example, if 20 messages are currently in the mailbox, 10 are of type 3, 7 are of type 16, and the rest are not of any recognized type, the following properties and counts are carried with the notification:</p> <pre>numMsgs=20 numMsgs3=10 numMsgs16=7</pre>
numSeen	MQInt32	<p>The number of messages in the mailbox marked as seen (read).</p> <p>This number counts the messages read by the mailbox owner. If other users have access to the mailbox, their actions in the mailbox are not included in this count. (However, the other users' actions can trigger notifications such as ReadMsg).</p>

TABLE 22–4 Properties Specific to Particular Notification Messages (Continued)

Property	Data Type	Description
numSeen nn	MQInt32	<p>The total number of messages in the mailbox marked as seen (read), specified for each message type. If message types are configured, a numSeennn property carries a count for each message type nn.</p> <p>The numSeen property is always sent; it counts the total number of all messages marked as seen, including all types.</p> <p>For example, if 20 messages are marked as seen, 10 are of type 3, 7 are of type 16, and the rest are not of any recognized type, the following properties and counts are carried with the notification:</p> <pre>numSeen=20 numSeen3=10 numSeen16=7</pre>
numSeenDeleted	MQInt32	<p>The number of messages in the mailbox marked as seen (read) and marked as deleted.</p> <p>This number counts the messages marked as read and deleted by the mailbox owner. If other users have access to the mailbox, their actions in the mailbox are not included in this count. (However, the other users' actions can trigger notifications such as ReadMsg and DeleteMsg).</p>
numSeenDeleted nn	MQInt32	<p>The total number of messages in the mailbox marked as seen (read) and marked as deleted, specified for each message type. If message types are configured, a numSeenDeletednn property carries a count for each message type nn.</p> <p>The numSeenDeleted property is always sent; it counts the total number of all messages marked as seen and deleted, including all types.</p> <p>For example, if 20 messages are marked as seen and deleted, 10 are of type 3, 7 are of type 16, and the rest are not of any recognized type, the following properties and counts are carried with the notification:</p> <pre>numSeenDeleted=20 numSeenDeleted3=10 numSeenDeleted16=7</pre>

TABLE 22–4 Properties Specific to Particular Notification Messages *(Continued)*

Property	Data Type	Description
oldflags	ConstMQString	The flags set for the user's mailbox message before they were changed by the current operation. This property is always present, together with newflags, when a <code>MsgFlags</code> notification message is produced. For the syntax and values for <code>oldflags</code> , see “ Syntax for newflags and oldflags Properties ” on page 710, below this table.
quotaRoot	ConstMQString	This can be a user name, folder name, or message type.
size	MQInt32	The size of the message. Note that this may not be the size of message body, since the body is typically a truncated version of the message.
uidValidity	MQInt32	The IMAP uid validity property.
userid	ConstMQString	The userid associated with the message.

Note – Subscribers should allow for undocumented properties when parsing the message reference. This allows for future compatibility when new properties are added.

Syntax for newflags and oldflags Properties

The `newflags` and `oldflags` properties are 5-character strings. The string must have the following values:

- If the /answered flag is set, the first character is "A". If not, it is blank (" ").
- If the /flagged flag is set, the second character is "F". If not, it is blank (" ").
- If the /deleted flag is set, the third character is "D". If not, it is blank (" ").
- If the /seen flag is set, the fourth character is "S". If not, it is blank (" ").
- If the /draft flag is set, the fifth character is "R". If not, it is blank (" ").

22.3.5.3

Properties Carried with Each Notification Message

[Table 22–5](#) shows the properties associated with each notification message.

For example, to see which properties apply to a `TrashMsg` message, look in the column header for “`ReadMsg`, `TrashMsg`.
A `TrashMsg` message can use `mailboxName`, `numMsgs`, `uidValidity`, `numSeen`, and `numDeleted` (in addition to the standard properties).

TABLE 22-5 Properties Carried with Each Notification Message

Property	NewMsg, UpdateMsg	ReadMsg, TrashMsg	DeleteMsg, PurgeMsg	MsgFlags	Login, Logout	OverQuota, UnderQuota
client	No	No	No	No	Yes	No
diskquota	No	No	No	No	No	Yes
diskquotauised	No	No	No	No	No	Yes
hdrLen	Yes	No	No	Yes	No	No
hostname	Yes	Yes	Yes	Yes	Yes	Yes
imapUid	Yes	No	Yes	Yes	No	No
lastUid	No	No	Yes	No	No	No
mailboxName	Yes	Yes	Yes	Yes	No	No
msgquota	No	No	No	No	No	Yes
newflags	No	No	No	Yes	No	No
numDeleted	Yes	Yes	Yes	No	No	No
numDeletedn	Yes*	Yes*	Yes*	No	No	No
numMsgs	Yes	Yes	Yes	No	No	Yes
numMsgsn	Yes*	Yes*	Yes*	No	No	No
numSeen	Yes	Yes	Yes	No	No	No
numSeenn	Yes*	Yes*	Yes*	No	No	No
numSeenDeleted	Yes	Yes	Yes	No	No	No
numSeenDeletedn	Yes*	Yes*	Yes*	No	No	No
oldflags	No	No	No	Yes	No	No
Owner	No	Yes	No	No	No	No
pid	Yes	Yes	Yes	Yes	Yes	Yes
process	Yes	Yes	Yes	Yes	Yes	Yes
quotaRoot	No	No	No	No	No	Yes
size	Yes	No	No	No	No	No
timestamp	Yes	Yes	Yes	Yes	Yes	Yes
uidValidity	Yes	Yes	Yes	Yes	No	No

TABLE 22-5 Properties Carried with Each Notification Message (Continued)

Property	NewMsg, UpdateMsg	ReadMsg, TrashMsg	DeleteMsg, PurgeMsg	MsgFlags	Login, Logout	OverQuota, UnderQuota
userid	No	Yes	No	No	Yes	Yes

Note – * The numDeletedn, numMsgsn, numSeenn, and numSeenDeletedn properties are carried with notifications only if message types are defined in the message store.

Configuring Security and Access Control

Messaging Server supports a full range of flexible security features that allow you to keep messages from being intercepted, prevent intruders from impersonating your users or administrators, and permit only specific people access to specific parts of your messaging system.

The Messaging Server security architecture is part of the security architecture of Sun Java System servers as a whole. It is built on industry standards and public protocols for maximum interoperability and consistency.

This chapter contains the following sections:

- “[23.1 About Server Security](#)” on page 713
- “[23.2 About HTTP Security](#)” on page 714
- “[23.3 Configuring Authentication Mechanisms](#)” on page 715
- “[23.4 User Password Login](#)” on page 719
- “[23.5 Configuring Encryption and Certificate-Based Authentication](#)” on page 720
- “[23.6 Configuring Administrator Access to Messaging Server](#)” on page 735
- “[23.7 Configuring Client Access to POP, IMAP, and HTTP Services](#)” on page 737
- “[23.8 Enabling POP Before SMTP](#)” on page 746
- “[23.9 Configuring Client Access to SMTP Services](#)” on page 749
- “[23.10 User/Group Directory Lookups Over SSL](#)” on page 749

23.1 About Server Security

Server security encompasses a broad set of topics. In most enterprises, ensuring that only authorized people have access to the servers, that passwords or identities are not compromised, that people do not misrepresent themselves as others when communicating, and that communications can be held confidential when necessary are all important requirements for a messaging system.

Perhaps because the security of server communication can be compromised in many ways, there are many approaches to enhancing it. This chapter focuses on setting up encryption, authentication, and access control. It discusses the following security-related Messaging Server topics:

- **User ID and password login:** requiring users to enter their user IDs and passwords to log in to IMAP, POP, HTTP, or SMTP, and the use of SMTP password login to transmit sender authentication to message recipients.
- **Encryption and authentication:** setting up your server to use the TLS and SSL protocols to encrypt communication and authenticate clients.
- **Administrator access control:** using the access-control facilities to delegate access to a Messaging Server and some of its individual tasks.
- **TCP client access control:** using filtering techniques to control which clients can connect to your server's POP, IMAP, HTTP, and authenticated SMTP services.

Not all security and access issues related to Messaging Server are treated in this chapter. Security topics that are discussed elsewhere include the following:

- **Physical security:** Without provisions for keeping server machines physically secure, software security can be meaningless.
- **Message-store access:** You can define a set of message-store administrators for the Messaging Server. These administrators can view and monitor mailboxes and can control access to them. For details, see [Chapter 20, “Managing the Message Store”](#)
- **End-user account configuration:** End-user account information can be primarily maintained by using the Delegated Administrator product.
- **Filtering unsolicited bulk email (UBE):** See [Chapter 18, “Mail Filtering and Access Control”](#)
- Secure Multipurpose Internet Mail Extensions (S/MIME) is described in [Chapter 24, “Administering S/MIME for Communications Express Mail.”](#)

There are a large number of documents that cover a variety of security topics. For additional background on the topics mentioned here and for other security-related information, see documentation web site at <http://docs.sun.com>.

23.2 About HTTP Security

Messaging Server supports user ID/password authentication, client certificate authentication, and Access Manager. There are some differences, however, in how the protocols handle network connections between client and server.

When a POP, IMAP, or SMTP client logs in to Messaging Server, a connection is made and a session is established. The connection lasts for the duration of the session; that is, from login to logout. When establishing a new connection, the client must reauthenticate to the server.

When an HTTP client logs in to Messaging Server, the server provides a unique session ID to the client. The client uses the session ID to establish multiple connections during a session. The HTTP client need not reauthenticate for each connection; the client need only reauthenticate if the session is dropped and the client wants to establish a new session. (If an HTTP session remains idle for a specified time period, the server will automatically drop the HTTP session and the client is logged out; the default time period is 2 hours.)

The following techniques are used to improve the security of HTTP sessions:

- The session IDs are bound to a specific IP address.
- Each session ID has a timeout value associated with it; if the session ID is not used for a specified time period, the session ID becomes invalid.
- The server keeps a database of all open session IDs, so a client cannot forge an ID.
- The session ID is stored in the URL, but not in any cookie files.

For information about specifying configuration parameters for improved connection performance, see [Chapter 5, “Configuring POP, IMAP, and HTTP Services”](#)

For information on Access Manager see [Chapter 6, “Enabling Single Sign-On \(SSO\)”](#)

23.3 Configuring Authentication Mechanisms

An authentication mechanism is a particular method for a client to prove its identity to a server. Messaging Server supports authentication methods defined by the Simple Authentication and Security Layer (SASL) protocol and it supports certificate-based authentication. The SASL mechanisms are described in this section. For more information about certificate-based authentication, see [“23.5 Configuring Encryption and Certificate-Based Authentication” on page 720](#).

Messaging Server supports the following SASL authentication methods for password-based authentication.

- **PLAIN** - This mechanism passes the user’s plaintext password over the network, where it is susceptible to eavesdropping.

Note that SSL can be used to alleviate the eavesdropping problem. For more information, see [“23.5 Configuring Encryption and Certificate-Based Authentication” on page 720](#)

- **DIGEST-MD5** - A challenge/response authentication mechanism defined in RFC 2831. (DIGEST-MD5 is not yet supported by Messaging Multiplexor.)

Note – This feature is deprecated and will be removed in a future release.

- **CRAM-MD5** - A challenge/response authentication mechanism similar to APOP, but suitable for use with other protocols as well. Defined in RFC 2195.

- **APOP** - A challenge/response authentication mechanism that can be used only with the POP3 protocol. Defined in RFC 1939.
- **LOGIN** - This is equivalent to PLAIN and exists only for compatibility with pre-standard implementations of SMTP authentication. By default the mechanism is only enabled for use by SMTP.

With a challenge/response authentication mechanism, the server sends a challenge string to the client. The client responds with a hash of that challenge and the user's password. If the client's response matches the server's own hash, the user is authenticated. The hash isn't reversible, so the user's password isn't exposed when sent over the network.

Note – The POP, IMAP, and SMTP services support all SASL mechanisms. The HTTP service supports only the plaintext password mechanism.

Table 23–1 shows some SASL and SASL-related configutil parameters. For the latest and most complete listing of configutil parameters, see the “[configutil Parameters](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

TABLE 23–1 Some SASL and SASL-related configutil Parameters

Parameter	Description
sasl.default.ldap.has_plain_passwords	Boolean to indicate that directory stores plaintext passwords which enables APOP, CRAM-MD5 and DIGEST-MD5. Default: False
sasl.default.transition_criteria	No longer supported or used. See sasl.default.auto_transition.
sasl.default.auto_transition	Boolean. When set and a user provides a plain text password, the password storage format will be transitioned to the default password storage method for the directory server. This can be used to migrate from plaintext passwords to APOP, CRAM-MD5 or DIGEST-MD5. Default: False
service imap allowanonymouslogin	This enables the SASL ANONYMOUS mechanism for use by IMAP. Default: False

TABLE 23-1 Some SASL and SASL-related configutil Parameters *(Continued)*

Parameter	Description
service.{imap pop http}.plainTextMinCipher	If this is > 0, then disable use of plaintext passwords unless a security layer (SSL or TLS) is activated. This forces users to enable SSL or TLS on their client to login which prevents exposure of their passwords on the network. The MMP has an equivalent option "RestrictPlainPasswords". NOTE: the 5.2 release of messaging server would actually check the value against the strength of the cipher negotiated by SSL or TLS. That feature has been eliminated to simplify this option and better reflect common-case usage. Default: 0
sasl.default.mech_list	A space-separated list of SASL mechanisms to enable. If non-empty, this overrides the sasl.default.ldap.has_plain_passwords option as well as the service imap.allowanonymouslogin option. This option applies to all protocols (imap, pop, smtp). Default: False
sasl.default.ldap.search_filter	This is the default search filter used to look up users when one is not specified in the inetDomainSearchFilter for the domain. The syntax is the same as inetDomainSearchFilter (see schema guide). Default: (&(uid=%U)(objectclass=inetmailuser))
sasl.default.ldap.searchfordomain	By default, the authentication system looks up the domain in LDAP following the rules for domain lookup (ref. needed) then looks up the user. However, if this option is set to "0" rather than the default value of "1", then the domain lookup does not happen and a search for the user (using the sasl.default.ldap.searchfilter) occurs directly under the LDAP tree specified by local.ugldapbasedn. This is provided for compatibility with legacy single-domain schemas, but use is not recommended for new deployments as even a small company may go through a merger or name change which requires support for multiple domains.

23.3.1 To Configure Access to Plaintext Passwords

To work, the CRAM-MD5, DIGEST-MD5, or APOP SASL authentication methods require access to the users' plaintext passwords. You need to perform the following steps:

1. Configure Directory Server to store passwords in cleartext.
2. Configure Messaging Server so that it knows Directory Server is using cleartext passwords.

▼ To Configure Directory Server to Store Cleartext Passwords

To enable CRAM-MD5, DIGEST-MD5, or APOP mechanisms, you must configure the Directory Server to store passwords in cleartext. If you are using a Directory Server prior to

version 6 the following instructions should apply. For version 6 or later, refer to the latest Directory Server documentation (*Sun Java System Directory Server Enterprise Edition 6.0 Administration Guide*): :

- 1 In the Directory Server Console, open the Directory Server you want to configure.
- 2 Click the Configuration tab.
- 3 Open Data in the left pane.
- 4 Click Passwords in the right pane.
- 5 From the Password encryption drop-down list, choose “cleartext”.

Note – This change only impacts users created in the future. Existing users will have to transition or have their password reset after this change.

23.3.1.1

To Configure Messaging Server for Cleartext Passwords

You can now configure Messaging Server so that it knows the Directory Server is able to retrieve cleartext passwords. This makes it safe for Messaging Server to advertise APOP, CRAM-MD5, and DIGEST-MD5:

```
configutil -o sasl.default.ldap.has_plain_passwords -v 1
```

You can disable these challenge/response SASL mechanisms by setting the value to 0.

Note – Existing users cannot use APOP, CRAM-MD5, or DIGEST-MD5 until their password is reset or migrated (see to Transition Users).

Note that MMP has an equivalent option: CRAMs.

23.3.2

To Transition Users

You can use configutil to specify information about transitioning users. An example would be if a user password changes or if a client attempts to authenticate with a mechanism for which they do not have a proper entry.

```
configutil -o sasl.default.auto_transition -v value
```

For value, you can specify one of the following:

- no or 0 - Don't transition passwords. This is the default.
- yes or 1 - Do transition passwords.

To successfully transition users, you must set up ACIs in the Directory Server that allow Messaging Server write access to the user password attribute. To do this, perform the following steps:

▼ To Transition Users

. If you are using a Directory Server prior to version 6 the following instructions apply. For version 6 or later, refer to the latest Directory Server documentation (*Sun Java System Directory Server Enterprise Edition 6.0 Administration Guide*):

- 1 In Console, open the Directory Server you want to configure.
- 2 Click the Directory tab.
- 3 Select the base suffix for the user/group tree.
- 4 From the Object menu, select Access Permissions.
- 5 Select (double click) the ACI for “Messaging Server End User Administrator Write Access Rights”.
- 6 Click ACI Attributes.
- 7 Add the userpassword attribute to the list of existing attributes.
- 8 Click OK.

`sasl.default.mech_list` can be used to enable a list of SASL mechanisms. If non-empty, this overrides the `sasl.default.ldap.has_plain_passwords` option as well as the `service imap.allowanonymouslogin` option. This option applies to all protocols (imap, pop, smtp).

23.4 User Password Login

Requiring password submission on the part of users logging into Messaging Server to send or receive mail is a first line of defense against unauthorized access. Messaging Server supports password-based login for its IMAP, POP, HTTP, and SMTP services.

23.4.1 IMAP, POP, and HTTP Password Login

By default, internal users must submit a password to retrieve their messages from Messaging Server. You enable or disable password login separately for POP, IMAP, and HTTP services. For more information about password login for POP IMAP, and HTTP Services, see “[5.2.2 Password-Based Login](#)” on page 141.

User passwords can be transmitted from the user's client software to your server as cleartext or in encrypted form. If both the client and your server are configured to enable SSL and both support encryption of the required strength (as explained in “[23.5.2 To Enable SSL and Selecting Ciphers](#)” on page 731), encryption occurs.

User IDs and passwords are stored in your installation's LDAP user directory. Password security criteria, such as minimum length, are determined by directory policy requirements; they are not part of Messaging Server administration.

Certificate-based login is an alternative to password-based login. It is discussed in this chapter along with the rest of SSL; see “[23.5.3 To Set Up Certificate-Based Login](#)” on page 733

Challenge/response SASL mechanisms are another alternative to plaintext password login.

23.4.2 SMTP Password Login

By default, users need not submit a password when they connect to the SMTP service of Messaging Server to send a message. You can, however, enable password login to SMTP in order to enable authenticated SMTP.

Authenticated SMTP is an extension to the SMTP protocol that allows clients to authenticate to the server. The authentication accompanies the message. The primary use of authenticated SMTP is to allow local users who are travelling (or using their home ISP) to submit mail (relay mail) without creating an open relay that others can abuse. The “AUTH” command is used by the client to authenticate to the server.

For instructions on enabling SMTP password login (and thus Authenticated SMTP), see “[12.4.4 SMTP Authentication, SASL, and TLS](#)” on page 376.

You can use Authenticated SMTP with or without SSL encryption.

23.5 Configuring Encryption and Certificate-Based Authentication

This section contains the following subsections:

- “[23.5.1 Obtaining Certificates](#)” on page 722
- “[23.5.2 To Enable SSL and Selecting Ciphers](#)” on page 731
- “[23.5.3 To Set Up Certificate-Based Login](#)” on page 733
- “[23.5.4 How to Optimize SSL Performance Using the SMTP Proxy](#)” on page 734

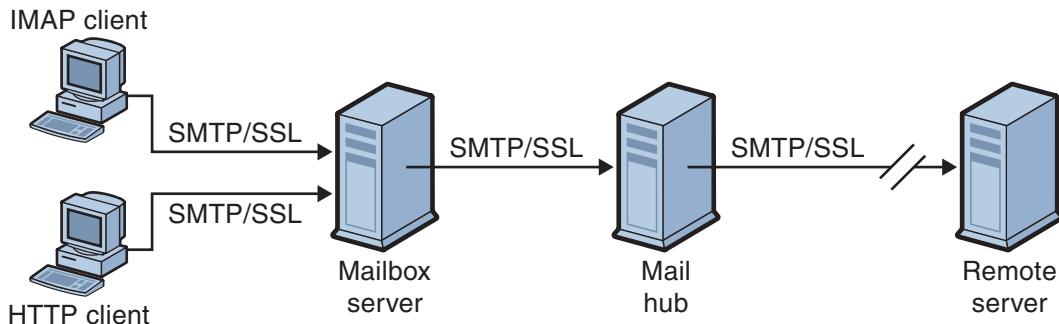
Messaging Server uses the Transport Layer Security (TLS) protocol, otherwise known as the Secure Sockets Layer (SSL) protocol, for encrypted communications and for certificate-based authentication of clients and servers. Messaging Server supports SSL versions 3.0 and 3.1. TLS is fully compatible with SSL and includes all necessary SSL functionality.

For background information on SSL, see *Introduction to SSL in Managing Servers With iPlanet Console 5.0*. SSL is based on the concepts of public-key cryptography, described in *Introduction to Public-Key Cryptography in Managing Servers With iPlanet Console 5.0*).

If transmission of messages between a Messaging Server and its clients and between the server and other servers is encrypted, there is little chance for eavesdropping on the communications. If connecting clients are authenticated, there is little chance for intruders to impersonate (spoof) them.

SSL functions as a protocol layer beneath the application layers of IMAP4, HTTP, POP3, and SMTP. SMTP and SMTP/SSL use the same port; HTTP and HTTP/SSL require different ports; IMAP and IMAP/SSL, and POP and POP/SSL can use the same port or different ports. SSL acts at a specific stage of message communication, as shown in [Figure 23–1](#), for both outgoing and incoming messages.

A. Outgoing message



B. Incoming message

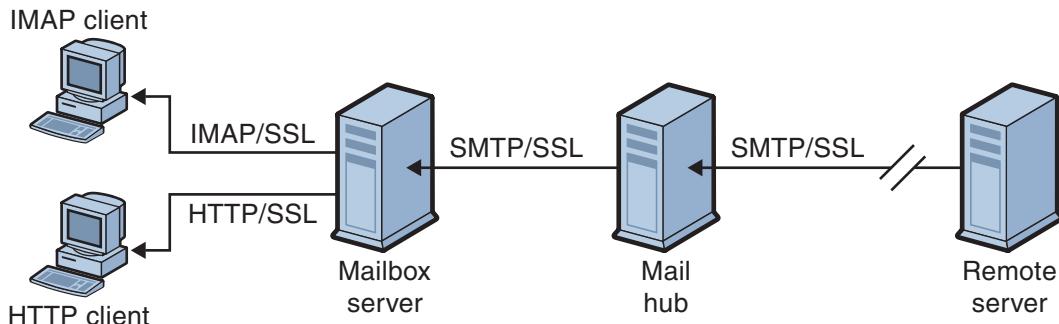


FIGURE 23–1 Encrypted Communications with Messaging Server

SSL provides hop-to-hop encryption, but the message is not encrypted on each intermediate server.

Note – To enable encryption for outgoing messages, you must modify the channel definition to include the `tls` channel keywords, such as `maytls`, `musttls`, and so on. For more information, see the “[12.4.8 Transport Layer Security](#)” on page [379](#) Manual.

Keep in mind that the extra overhead in setting up an SSL connection can put a performance burden on the server. In designing your messaging installation and in analyzing performance, you may need to balance security needs against server capacity.

23.5.1 Obtaining Certificates

Whether you use SSL for encryption or for authentication, you need to obtain a server certificate for your Messaging Server. The certificate identifies your server to clients and to other servers. Another way of obtaining certificates is using `msgcert` command (described later in this section). Note that the old `certutil` command still works, but may be more complex to use and is not internationalized. For more information on `certutil`, see “[23.5 Configuring Encryption and Certificate-Based Authentication](#)” on page [720](#) and <http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html>.

This section consists of the following subsections:

- “[23.5.1.1 To Manage Internal and External Modules](#)” on page [722](#)
- “[23.5.1.2 Creating a Password File](#)” on page [723](#)
- “[23.5.1.3 Obtaining and Managing Certificates](#)” on page [724](#)
- “[23.5.1.4 About msgcert](#)” on page [724](#)
- “[23.5.1.5 Managing Certificates](#)” on page [725](#)
- “[To Create a Messaging Server Certificate Database with a Default Self-signed Certificate](#)” on page [726](#)
- “[To Manage Self-Signed Certificates](#)” on page [726](#)
- “[23.5.1.6 To Install Certificates of Trusted CAs](#)” on page [726](#)

23.5.1.1 To Manage Internal and External Modules

A server certificate establishes the ownership and validity of a key pair, the numbers used to encrypt and decrypt data. Your server’s certificate and key pair represent your server’s identity. They are stored in a certificate database that can be either internal to the server or on an external, removable hardware card (smartcard).

Sun Java System servers access a key and certificate database using a module conforming to the Public-Key Cryptography System (PKCS) #11 API. The PKCS #11 module for a given hardware device is usually obtained from its supplier and must be installed into the Messaging Server

before the Messaging Server can use that device. The pre-installed “Netscape Internal PKCS # 11 Module” supports a single internal software token that uses the certificate database that is internal to the server.

Setting up the server for a certificate involves creating a database for the certificate and its keys and installing a PKCS #11 module. If you do not use an external hardware token, you create an internal database on your server, and you use the internal, default module that is part of Messaging Server. If you do use an external token, you connect a hardware smartcard reader and install its PKCS #11 module.

Note – In the following sections we will refer to the console or Directory Server Console. This refers to Directory Server prior to version 6. For version 6 or later, the graphical user interface is called the Directory Server Control Center. Refer to the latest Directory Server documentation (*Sun Java System Directory Server Enterprise Edition 6.0 Administration Guide*) for more information.

You can manage PKCS #11 modules, whether internal or external, through Console. To install a PKCS #11 module:

1. Connect a hardware card reader to the Messaging Server host machine and install drivers.
2. Use the modutil found in `msg-svr-base/sbin` to install the PKCS #11 module for the installed driver.

Installing Hardware Encryption Accelerators. If you use SSL for encryption, you may be able to improve server performance in encrypting and decrypting messages by installing a hardware encryption accelerator. An encryption accelerator typically consists of a hardware board, installed permanently in your server machine, plus a software driver. Messaging Server supports accelerator modules that follow the PKCS #11 API. (They are essentially hardware tokens that do not store their own keys; they use the internal database for that.) You install an accelerator by first installing the hardware and drivers as specified by the manufacturer, and then completing the installation—as with hardware certificate tokens—by installing the PKCS #11 module.

23.5.1.2

Creating a Password File

On most Sun Java System servers for which SSL is enabled, the administrator is prompted at startup to supply the password required to decrypt the key pair. On Messaging Server, however, to alleviate the inconvenience of having to enter the password multiple times (it is needed by at least three server processes), and to facilitate unattended server restarts, the password is read from a password file. Passwords themselves are generated when their certificate database is created using the `msgcert generate_certdb` command.

The password file is named `sslpassword.conf` and is in the directory `msg-svr-base/config/`. Entries in the file are individual lines with the format

moduleName:password

where *moduleName* is the name of the (internal or external) PKCS #11 module to be used, and *password* is the password that decrypts that module's key pair. The password is stored as clear (unencrypted) text.

Messaging Server provides a default version of the password file, with the following single entry (for the internal module and default password):

Internal (Software) Token:netscape!

If you specify anything but the default password when you install an internal certificate, you need to edit the above line of the password file to reflect the password you specified. If you install an external module, you need to add a new line to the file, containing the module name and the password you specified for it.



Caution – Because the administrator is not prompted for the module password at server startup, it is especially important that you ensure proper administrator access control to the server and proper physical security of the server host machine and its backups.

23.5.1.3

Obtaining and Managing Certificates

Whether you use SSL for encryption or for authentication, you need to obtain a server certificate for your Messaging Server. The certificate identifies your server to clients and to other servers. The primary mechanism for obtaining and managing certificates is using the `msgcert`. However, if the Administration Server is installed, you can also use the Administration Console.

The rest of this section describes how to use `msgcert`.

23.5.1.4

About `msgcert`

`msgcert` allows you to generate a certificate request, add a certificate to the certificate database, list certificates in the database and so on. For detailed information enter the following at the command line:

`msg-svr-base/sbin/msgcert --help`

This is shown below.

```
# ./msgcert --help
```

Usage: `msgcert SUBCMD [GLOBAL_OPTS] [SUBCMD_OPTS] [SUBCMD_OPERANDS]`

Manages the Messaging Servers Certificate Database

The accepted values for SUBCMD are:

add-cert	Adds a certificate to the certificate database
add-selfsign-cert	Creates and adds a selfsign certificate to the certificate database
export-cert	Exports a certificate and its keys from the database
generate-certDB	Creates Messaging Server Databases cert8.db key3.db secmod.db and sslPassword
import-cert	Adds a new certificate and its keys to the cert database
import-selfsign-cert	Adds a new selfsign certificate and its keys to the cert database
list-certs	Lists all certificates in the Certificate database
remove-cert	Removes a certificate from the database
renew-cert	Renews a certificate
renew-selfsign-cert	Renews a selfsign certificate
request-cert	Generates a certificate request
show-cert	Displays a certificate

The accepted value for GLOBAL_OPTS is:-?, --help
Displays SUBCMD help

NOTE: You must stop all the TLS or SSL-enabled servers before making any changes to the Certificate Database.

Each of the sub-commands shown above performs a specific certificate management function. Details about these sub-commands and their functions can be obtained by entering the following:

`msgcert SUBCMD --help`

The remainder of this section will describe some common certificate management procedures.

23.5.1.5

Managing Certificates

This section describes how to manage SSL certificates in Messaging Server. To run SSL on Messaging Server, you must either use a self-signed certificate or a Public Key Infrastructure (PKI) solution which involves an external Certificate Authority (CA). For a PKI solution, you need a CA-signed server certificate which contains both a public and a private key. This certificate is specific to one Messaging Server. You also need a trusted CA certificate, which contains a public key. The trusted CA certificate ensures that all server certificates from your CA are trusted. This certificate is sometimes also called a CA root key or root certificate.

Configuring the Certificate Database Password

When managing certificates, you do not need to type a certificate password or specify the password file. You can simply pass the password as -W argument. Example:

```
echo "password22" > /tmp/certdbpwd
echo "password22" > /tmp/certdbpwd
# ./msgcert list-certs -W /tmp/certdbpwd
```

▼ To Create a Messaging Server Certificate Database with a Default Self-signed Certificate

- 1 To create a Messaging Server Certificate Database run the following command:

```
msgcert generate-certDB
```

This reads the certificate database password from CERT_PW_FILE (Default: prompt for password)

- 2 You can view this certificate by using the command:

```
msgcert show-cert Server-Cert
```

▼ To Manage Self-Signed Certificates

If you are using certificates for test purposes, you can use self-signed certificates. In deployment configurations, you might prefer to use trusted Certificate Authority (CA) certificates. You can also use the Directory Server Admin Console to perform this task.

- 1 When you create the certificate database, a default self-signed certificate is automatically provided. If you want to use a self-signed certificate with non-default settings, use the msgcert add-selfsign-cert command. Example:

```
msgcert add-selfsign-cert --name siroe --org comms --org-unit Messaging  
--city SantaClara --state ca --country us MySelfSigned-Cert
```

A self-signed certificate is valid for three months.

- 2 When your self-signed certificate expires, renew the certificate with the command:

```
msgcert renew-selfsign-cert cert_alias
```

23.5.1.6

To Install Certificates of Trusted CAs

Use ./msgcert add-cert to install the certificates of certificate authorities. A CA certificate validates the identity of the CA itself. The server uses these CA certificates in the process of authenticating clients and other servers.

If, for example, you set up your enterprise for certificate-based client authentication in addition to password-based authentication (see “Setting Up Certificate-Based Login” on page 157), you need to install the CA certificates of all CAs that are trusted to issue the certificates that your clients may present. These CAs may be internal to your organization or they may be external, representing commercial or governmental authorities or other enterprises. (For more details on the use of CA certificates for authentication, see *Introduction to Public-Key Cryptography in Managing Servers With iPlanet Console 5.0*.)

When installed, Messaging Server initially contains CA certificates for several commercial CAs. If you need to add other commercial CAs or if your enterprise is developing its own CA for internal use (using Sun Java System Certificate Server), you need to obtain and install additional CA certificates.

Note – The CA certificates automatically provided with Messaging Server are not initially marked as trusted for client certificates. You need to edit the trust settings if you want to trust client certificates issued by these CAs. For instructions, see “[23.5.1 Obtaining Certificates](#)” on page [722](#).

The following procedure describes the process of requesting and installing CA-signed server and trusted CA certificates for use with Messaging Server.

▼ To Request a CA-Signed Server Certificate

You can also use the Directory Server Admin Console to perform this task.

1 Generate a CA-signed server certificate request.

```
msgcert request-cert [-W CERT_PW_FILE] {-S DN|--name NAME [--org ORG] [--org-unit ORG-UNIT]
[--city CITY] [--state STATE] [--country COUNTRY] } [-F FORMAT] [-o OUTPUT_FILE]
```

Below is an example of a request for a CA-signed server certificate. It returns the certificate in binary format:

```
./msgcert request-cert --name aqua --org siroe --org-unit Messaging -o my_ca_signed_request_cert
```

To return the certificate in ASCII format use the command as follows:

```
./msgcert request-cert --name aqua --org siroe --org-unit Messaging -F ascii -o my_casigned_request_cert
```

Certificate Authorities usually require all of the attributes that are shown in this example in order to completely identify the server. For a description of each attribute, enter `./msgcert request-cert --help`. When you request a certificate by using `msgcert request-cert`, the resulting certificate request is a binary certificate request unless you specify ASCII as output format. If you specify ASCII, the resulting certificate request is a PKCS #10 certificate request in PEM format. PEM is the Privacy Enhanced Mail format specified by RFCs 1421 through 1424 and used to represent a base64-encoded certificate request in US-ASCII characters. The content of the request look similar to the following example:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBdT...9l
MRAwDgYDVQQDEwdhcXVhdGljMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQD
KEh5Fnj/h9GEu18Da6DKJpcNShkwxanjnKs2883ZoUV5Sp4pN7U6Vfbh0414WXZh
D26m3t81q9b9h47Klkf0pW1X3BB6L0jGOHSt2VoNBI8n3hJ6XiN2zYbrLTgdKuo
y0YrSG/kHFnqKghikag90/Ox+cwD+mpjl2QnsPZgswIDAQABoAAwDQYJKoZIhvcN
```

```

AQEEBQADgYEArqgWQIwNZDC2d3EZawI23Wj9o6Pyvu9J1rkb+NYgIEnNp9jugxqX
F326N0ABLdHXXNX/2ZvC5TK0gS4RidTBM89N9xJvokmvRGfc+1x80uxy474YdNlZ
s+nP8AYo9dw9mrL0ammozx9HLPsvYNFp4FxekgV2n8QG7WC5rkN5bCE=
-----END NEW CERTIFICATE REQUEST-----

```

2 Transmit the certificate request to your Certificate Authority, according to its procedures.

The process for obtaining your Certificate Authority certificate differs depending on the certificate authority you use. Some commercial CAs provide a website that allows you to automatically download the certificate. Other CAs will email it to you upon request.

After you have sent your request, you must wait for the CA to respond with your certificate. Response time for your request varies. For example, if your CA is internal to your company, the CA might only take a day or two to respond to your request. If your selected CA is external to your company, the CA could take several weeks to respond to your request.

3 Save the certificate you receive back from the Certificate Authority.

You should back up your certificates in a safe location. If you ever lose the certificates, you can reinstall them by using your backup file. You can save them in text files. The PKCS #11 certificate in PEM format looks similar to the following example.

```

-----BEGIN CERTIFICATE-----
MIICjCCA ZugAwIBAgICCEEwDQYJKoZIhKqvNAQFBQAwfDELMKA1UEBhMCVVMx
IzAhBgNVBAoG1BhB9a2FwAxsZGwSBxwRnZXrLcBjbmMuMR0wGwYDVQQLExRX
aWRnZXQgTW3FrZXJzICdSJyBvczEpMCCGAx1UEAxgVGvzdCBUXN0IFRlc3QgVGvz
dCBUXN0IFlC3QgQ0EswHhcNOTgwMzEyMDIzMzUWhcNOTgwMzI2MDIzMpzU3WjBP
MQswCYDDVQQGEwJVUzEoMCYGA1UEChMfTmV0c2NhCGugRGlzN0b3J5VIFB1Ymxp
Y2F0alw9uczEWMB4QGA1UEAxMNZHvg49dq2tLNvbjTBaMA0GCSqGSIb3DQEBAQUA
A0kAMEYkCQCksMR/aLGdfp4m00iGgijG5KgOsryRNvwGYw7kfW+8mmijDtZaRjYnj
jcgpF3VnlbxbcLx9Lvj jNLC5737XzdAgEDozYwpNDARBglghkgBhvCEAQEEBAMC
APAwHkwYDVR0jBBgwFAU67URjwCaGqZHUpSpdLxlzwJKiMwDQYJKoZIhQvcNAQEF
BQADgYEAJ+BfVem3vBOPBveNdLGfjl b9hucgmaMcQa9FA/db8qimKT/ue9UGOJql
bwbMKBBopsDn56p2yV3PLIsBgrcuSoBCuFFnxBnqSiTS7YiYgCwqWaUA0ExJFmD6
6hBLseqkSWulk+hXHN7L/NrVi0+7zNtKcaZLlFPF7d7j2MgX4Bo=
-----END CERTIFICATE-----

```

▼ To Add the CA-Signed Server Certificate and Trusted CA Certificate

You can also use the Directory Server Admin Console to perform this task.

1 Add the CA-signed server certificate using the following command:

```
msgcert add-cert cert_alias cert_file
```

Where *cert_alias* is a name which you give to identify your certificate, and *cert_file* is the text file containing the PKCS #11 certificate in PEM format.

For example, to install a CA-signed server certificate, you might use a command similar to:

```
msgcert add-cert /my_cert/server-cert-file
```

The certificate is now installed, but is not yet trusted. To trust the CA-signed server certificate, you must install the Certificate Authority certificate.

2 Add the trusted Certificate Authority certificate using the following command:

```
msgcert add-cert -C cert_alias cert_file
```

The -C option indicates that the certificate is a trusted Certificate Authority certificate.

For example, to install a trusted certificate from a Certificate Authority, you might use the command:

```
msgcert add-cert -C CA-cert /my_cert/ca-cert-file
```

3 Optionally, use the following command to verify your installed certificates:

To list all server certificates, showing information such as alias and validity dates:

```
msgcert list-certs
```

The Messaging server will have a default certificate called Server-Cert when generated with ./msgcert generate-CertDB . The text Same as issuer indicates that the default certificate is a self-signed server certificate. For example:

```
# ./msgcert list-certs
Enter the certificate database password:
Alias          Valid from      Expires on      Self-signed      Issued by           Issued to
-----          -----          -----          -----          -----          -----
SelfSignedCrt  2006/07/28 12:58  2006/10/28 12:58  y      CN=SFO,L=SC,ST=ca,C=us  Same as issuer
Server-Cert    2006/07/28 07:47  2006/10/28 07:47  y      CN=perseids        Same as issuer
2 certificates found
```

To list trusted CA certificates:

```
msgcert list-certs -C
```

To view the details of a certificate, including the certificate expiry date:

```
msgcert show-cert cert_alias
```

For example, to show a self-signed certificate:

```
# ./msgcert show-cert MySelfSigned-Cert
Enter the certificate database password:
Certificate:
Data:
Version: 3 (0x2)
Serial Number:
00:83:35:37:94
Signature Algorithm: PKCS #1 MD5 With RSA Encryption
```

Issuer:
"CN=siroe,O=commes,OU=Messaging,L=SantaClara,ST=ca,C=us"
Validity:
Not Before: Fri Jul 28 19:58:31 2006
Not After : Sat Oct 28 19:58:31 2006
Subject:
"CN=siroe,O=commes,OU=Messaging,L=SantaClara,ST=ca,C=us"
Subject Public Key Info:
Public Key Algorithm: PKCS #1 RSA Encryption
RSA Public Key:
Modulus:
aa:9d:3d:23:b2:59:39:f3:77:c8:69:7f:b0:d1:ac:d2:
4e:81:c8:51:0f:27:6f:a1:21:4b:a9:27:46:d7:0f:b4:
c8:44:86:32:5e:4f:2f:1c:2f:a9:b8:a3:49:b5:b8:ab:
51:a8:a5:ba:1c:e8:90:7d:46:67:f9:a7:44:c5:1d:24:
e6:bd:e8:8f:07:b4:5a:68:41:b1:19:f2:ea:98:ba:25:
55:b8:ba:9c:af:bb:43:c3:c0:8f:14:a7:4c:2b:50:b4:
ac:df:b5:cd:68:de:a6:14:9d:68:77:d3:8b:7f:de:c0:
5d:35:d7:55:8d:b5:c3:14:2a:60:a9:bf:de:96:90:a9
Exponent: 65537 (0x10001)
Signature Algorithm: PKCS #1 MD5 With RSA Encryption
Signature:
15:86:f1:cc:85:c9:08:0f:ff:d3:56:d8:e2:c8:ea:3c:
8e:45:36:be:8b:b0:7d:2f:e9:cd:e3:b4:ad:8c:70:59:
c8:a5:14:da:9c:fa:7f:70:86:64:34:0b:21:ae:c4:28:
d2:f5:94:5c:a6:78:0f:d9:fd:fc:c5:5e:37:49:25:a9:
bc:12:59:cb:fb:4e:e9:d4:8a:8d:3d:41:12:ae:f1:7f:
8d:d3:10:ac:fb:33:51:5d:0c:1b:dc:23:5f:95:d5:6d:
c6:1d:e5:ed:13:8b:16:41:89:5b:4d:de:c0:c7:56:a2:
48:82:38:32:5a:99:d5:21:20:c5:0d:5c:ea:0c:84:aa
Fingerprint (MD5):
EF:76:A3:6C:09:4E:BC:6B:87:76:A3:35:70:1F:B2:C4
Fingerprint (SHA1):
BB:1C:20:4B:79:3A:F1:49:F0:83:FB:CC:9C:56:10:D3:06:97:AA:07

Certificate Trust Flags:
SSL Flags:
Valid CA
Trusted CA
User
Trusted Client CA
Email Flags:
User
Object Signing Flags:
User

▼ Renewing an Expired CA-Signed Server Certificate

When your CA-signed server certificate (public and private key) expires, you can renew it by using the following procedure. You can also use the Directory Server Admin Console to perform this task.

- 1 Obtain an updated CA-signed server certificate from your Certificate Authority.
- 2 Once you receive the updated certificate, install the certificate.

```
msgcert renew-cert cert_alias cert_file
```

▼ To Export and Import a CA-Signed Server Certificate

In some cases you might want to export a certificate so that you can later import the certificate, for example, to another host. You can also use the Directory Server Admin Console to perform this task.

- 1 Export the certificate.

```
msgcert export-cert [-o OUTPUT_FILE] CERT_ALIAS
```

For example:

```
$ ./msgcert export-cert -o /tmp/first-certificate "First Certificate"
$ ./msgcert export-cert -o /tmp/first-server-certificate Server-Cert
Choose the PKCS#12 file password:
Confirm the PKCS#12 file password:
$ls /tmp
first-server-certificate
/tmp/first-certificate
```

- 2 Import the certificate.

```
$ msgcert import-cert CERT_FILE
```

For example, to import the certificate

```
$ msgcert import-cert /tmp/first-server-certificate
Enter the PKCS#12 file password:
$
```

23.5.2 To Enable SSL and Selecting Ciphers

You can use Console to enable SSL and to select the set of encryption ciphers that Messaging Server can use in its encrypted communications with clients. You can also install the SSL certificate using the msgcert utility and run the appropriate configutil or edit the appropriate configuration files necessary to enable SSL for that particular service.

23.5.2.1

About Ciphers

A *cipher* is the algorithm used to encrypt and decrypt data in the encryption process. Some ciphers are stronger than others, meaning that a message they have scrambled is more difficult for an unauthorized person to unscramble.

A cipher operates on data by applying a key—a long number—to the data. Generally, the longer the key the cipher uses during encryption, the harder it is to decrypt the data without the proper decryption key.

When a client initiates an SSL connection with a Messaging Server, the client lets the server know what ciphers and key lengths it prefers to use for encryption. In any encrypted communication, both parties must use the same ciphers. Because there are a number of cipher-and-key combinations in common use, a server should be flexible in its support for encryption. Messaging Server can support up to 6 combinations of cipher and key length.

Table 23–2 lists the ciphers that Messaging Server supports for use with SSL 3.0. The table summarizes information that is available in more detail in the *Introduction to SSL* section of *Managing Servers with iPlanet Console*.

TABLE 23–2 SSL Ciphers for Messaging Server

Cipher	Description
RC4 with 128-bit encryption and MD5 message authentication	The fastest encryption cipher (by RSA) and a very high-strength combination of cipher and encryption key.
Triple DES with 168-bit encryption and SHA message authentication	A slower encryption cipher (a U.S. government-standard) but the highest-strength combination of cipher and encryption key.
DES with 56-bit encryption and SHA message authentication	A slower encryption cipher (a U.S. government-standard) and a moderate-strength combination of cipher and encryption key.
RC4 with 40-bit encryption and MD5 message authentication	The fastest encryption cipher (by RSA) and a lower-strength combination of cipher and encryption key.
RC2 with 40-bit encryption and MD5 message authentication	A slower encryption cipher (by RSA) and a lower-strength combination of cipher and encryption key.
No encryption, only MD5 message authentication	No encryption; use of a message digest for authentication alone.

Unless you have a compelling reason for not using a specific cipher, you should support them all. However, note that export laws restrict the use of certain encryption ciphers in certain countries. Also, some client software produced before the relaxation of United States Export Control laws cannot use the higher strength encryption. Be aware that while the 40-bit ciphers might hinder the casual eavesdropper, they are not secure and therefore will not stop a motivated attack.

To enable SSL and select encryption ciphers, follow these command line steps:

To specify a certificate:

```
configutil -o encryption.rsa.nsslpersonalityssl -v certname
```

There is also a per-service configuration setting for the SSL server certificate nickname. The new configutil settings are as follows:

`local.imta.sslnicknames` for the SMTP and Submit servers; `local imap.sslnicknames` for the IMAP server; `local.pop.sslnicknames` for the POP server; `local.http.sslnicknames` for web mail server.

They have the same meaning (and override) the `encryption.rsa.nsslpersonalityssl` setting. Specifically, this is a comma-separated list of NSS certificate nicknames. Although more than one nickname is permitted in the list, each nickname must refer to a different type of certificate (for example, an RSA cert and a DSS cert) so the setting will almost always be only one nickname. A nickname can be unqualified in which case the NSS software token or default token will be searched, or it can have the form `security-module:nickname`" in which case the specified security module will be searched for that nickname. This is needed for certificates stored in hardware tokens or places other than the default NSS database.

This does not permit the use of more than one NSS software token in the product. In particular, there is only one `cert8.db`, `key3.db` and `secmod.db` for IMAP, POP, SMTP and HTTP. NSS doesn't permit that.

Note – To enable SSL encryption for outgoing messages, you must modify the channel definition to include the `tls` channel keywords, such as `maytls`, `musttls`, and so on. For more information, see the “[12.4.8 Transport Layer Security](#)” on page 379 Manual.

23.5.3

To Set Up Certificate-Based Login

In addition to password-based authentication, Sun Java System servers support authentication of users through examination of their digital certificates. In certificate-based authentication, the client establishes an SSL session with the server and submits the user's certificate to the server. The server then evaluates whether the submitted certificate is genuine. If the certificate is validated, the user is considered authenticated.

To set up your Messaging Server for certificate-based login:

▼ To Set Up Certificate-Based Login

- 1 Obtain a server certificate for your server. (For details, see “[23.5.1 Obtaining Certificates](#)” on page 722)

- 2 Run the Certificate Setup Wizard to install the certificates of any trusted certificate authorities that will issue certificates to the users your server will authenticate. (For details, see “[23.5.1.6 To Install Certificates of Trusted CAs](#)” on page 726)

Note that as long as there is at least one trusted CA in the server’s database, the server requests a client certificate from each connecting client.

- 3 Turn on SSL. (For details, see “[23.5.2 To Enable SSL and Selecting Ciphers](#)” on page 731)
- 4 (Optional) Edit your server’s `certmap.conf` file so that the server appropriately searches the LDAP user directory based on information in the submitted certificates.

Editing the `certmap.conf` file is not necessary if the email address in your users’ certificates matches the email address in your users’ directory entries, and you do not need to optimize searches or validate the submitted certificate against a certificate in the user entry.

For details of the format of `certmap.conf` and the changes you can make, see the SSL chapter of *Managing Servers with iPlanet Console*.

Once you have taken these steps, when a client establishes an SSL session so that the user can log in to IMAP or HTTP, the Messaging Server requests the user’s certificate from the client. If the certificate submitted by the client has been issued by a CA that the server has established as trusted, and if the identity in the certificate matches an entry in the user directory, the user is authenticated and access is granted (depending on access-control rules governing that user).

There is no need to disallow password-based login to enable certificate-based login. If password-based login is allowed (which is the default state), and if you have performed the tasks described in this section, both password-based and certificate-based login are supported. In that case, if the client establishes an SSL session and supplies a certificate, certificate-based login is used. If the client does not use SSL or does not supply a certificate, the server requests a password.

23.5.4

How to Optimize SSL Performance Using the SMTP Proxy

Most sites should not use the SMTP proxy as it adds additional latency to the SMTP protocol. However, a large-scale site which makes heavy use of SSL to protect SMTP connections may wish to maximize their investment in SSL accelerator hardware by performing all SSL operations for all protocols on a server which does nothing other than SSL and proxy. The SMTP proxy allows SSL to be processed by a front end proxy server while the mail queues are on a separate MTA machine. This way hardware optimized for each task can be separately configured and purchased.

See “[Using the MMP SMTP Proxy](#)” in *Sun Java Communications Suite 5 Deployment Planning Guide* and “[23.8 Enabling POP Before SMTP](#)” on page 746 for instructions on how to install the SMTP Proxy.

23.6 Configuring Administrator Access to Messaging Server

This section mostly pertains to the Sun Java System LDAP Schema v. 1. This section contains the following subsections:

- “[23.6.1 Hierarchy of Delegated Administration](#)” on page 735
- “[To Provide Access to the Server as a Whole](#)” on page 736
- “[23.6.2 To Restrict Access to Specific Tasks](#)” on page 736

This section describes how to control the ways in which server administrators can gain access to Messaging Server. Administrative access to a given Messaging Server and to specific Messaging Server tasks occurs within the context of delegated server administration.

Delegated server administration is a feature of most Sun Java System servers; it refers to the capability of an administrator to provide other administrators with selective access to individual servers and server features. This chapter briefly summarizes delegated server tasks. For more detailed information, see the chapter on delegating server administration in *Managing Servers with iPlanet Console*.

23.6.1 Hierarchy of Delegated Administration

When you install the first Sun Java System server on your network, the installation program automatically creates a group in the LDAP user directory called the Configuration Administrators group. By default, the members of the Configuration Administrators group have unrestricted access to all hosts and servers on your network.

The Configuration Administrators group is at the top of an access hierarchy, such as the following, that you can create to implement delegated administration (if Sun Java System LDAP Schema v. 1 is used) for Messaging Server:

1. **Configuration administrator.** The “super user” for the network of Sun Java System servers. Has complete access to all resources.
2. **Server administrator.** A domain administrator might create groups to administer each type of server. For example, a Messaging Administrators group might be created to administer all Messaging Servers in an administrative domain or across the whole network. Members of that group have access to all Messaging Servers (but no other servers) in that administrative domain.
3. **Task administrator.** Finally, any of the above administrators might create a group, or designate an individual user, with restricted access to a single Messaging Server or a set of Messaging Servers. Such a task administrator is permitted to perform only specific, limited server tasks (such as starting or stopping the server only, or accessing logs of a given service).

Console provides convenient interfaces that allow an administrator to perform the following tasks:

- Grant a group or an individual access to a specific Messaging Server, as described in “Providing Access to the Server as a Whole” (next).
- Restrict that access to specific tasks on a specific Messaging Server, as described in “[23.6.2 To Restrict Access to Specific Tasks](#)” on page 736.

▼ **To Provide Access to the Server as a Whole**

This section describes to give a user or group permission to access a given instance of Messaging Server.

1 Log in to Console as an administrator with access to the Messaging Server you want to provide access to.

2 Select that server in the Console window.

From the Console menu, choose Object, then choose Set Access Permissions.

3 Add or edit the list of users and groups with access to the server.

(For more complete instructions, see the chapter on delegating server administration in *Managing Servers with iPlanet Console*.)

Once you have set up the list of individuals and groups that have access to the particular Messaging Server, you can then use ACIs, as described next, to delegate specific server tasks to specific people or groups on that list.

23.6.2 To Restrict Access to Specific Tasks

An administrator typically connects to a server to perform one or more administrative tasks. Common administrative tasks are listed in the Messaging Server Tasks form in Console.

By default, access to a particular Messaging Server means access to all of its tasks. However, each task in the Task form can have an attached set of access-control instructions (ACIs). The server consults those ACIs before giving a connected user (who must already be a user with access permissions to the server as a whole) access to any of the tasks. In fact, the server displays in the Tasks form only those tasks to which the user has permission.

If you have access to a Messaging Server, you can create or edit ACIs on any of the tasks (that is, on any of the tasks to which you have access), and thus restrict the access that other users or groups can have to them.

▼ To Restrict the Task Access of a User or Group

- 1 Log in to the Console as an administrator with access to the Messaging Server you want to provide restricted access to.
- 2 Open the server and select a task in the server's Tasks form by clicking on the Task text.
- 3 From the Edit menu, choose Set Access Permissions, and add or edit the list of access rules to give a user or group the kind of access you want them to have.
- 4 Repeat the process for other tasks, as appropriate.

(For more complete instructions, see the chapter on delegating server administration in *Managing Servers with iPlanet Console*.)

ACIs and how to create them are described more fully in the chapter on delegating server administration in *Managing Servers with iPlanet Console*.

23.7 Configuring Client Access to POP, IMAP, and HTTP Services

This section contains the following subsections:

- “23.7.1 How Client Access Filters Work” on page 738
- “23.7.2 Filter Syntax” on page 739
- “23.7.3 Filter Examples” on page 743
- “23.7.4 To Create Access Filters for Services” on page 745
- “23.7.5 To Create Access Filters for HTTP Proxy Authentication” on page 745

Messaging Server provides two methods of access control on a service-by-service basis for its IMAP, POP, and HTTP services. These processes allow you to exercise far-ranging and fine-grained control over which clients can gain access to your server. The first method allows you to specify control over specific internal users and domains. You do this by setting the LDAP attributes “[mailAllowedServiceAccess](#)” in *Sun Java Communications Suite 5 Schema Reference* or “[mailDomainAllowedServiceAccess](#)” in *Sun Java Communications Suite 5 Schema Reference*.

The second method, which is described in this section, uses the configutil parameters `service.service.domainallowed`, `service.service.domainnotallowed` and `service.service.proxydomainallowed`.

If you are managing messaging services for a large enterprise or an Internet service provider, these capabilities can help you to exclude spammers and DNS spoofers from your system and improve the general security of your network. For control of unsolicited bulk email specifically, see also [Chapter 18, “Mail Filtering and Access Control”](#)

Note – If controlling access by IP address is not an important issue for your enterprise, you do not have to create any of the filters described in this section. If minimal access control is all you need, see “[23.7.3.2 Mostly Allowing](#)” on page [743](#) for instructions on setting it up.

23.7.1 How Client Access Filters Work

The Messaging Server access-control facility is a program that listens at the same port as the TCP daemon it serves; it uses access filters to verify client identity, and it gives the client access to the daemon if the client passes the filtering process.

As part of its processing, the Messaging Server TCP client access-control system performs (when necessary) the following analyses of the socket end-point addresses:

- Reverse DNS lookups of both end points (to perform name-based access control)
- Forward DNS lookups of both end points (to detect DNS spoofing)
- Identd callback (to check that the user on the client end is known to the client host)

The system compares this information against access-control statements called *filters* to decide whether to grant or deny access. For each service, separate sets of Allow filters and Deny filters control access. Allow filters explicitly grant access; Deny filters explicitly forbid access.

When a client requests access to a service, the access-control system compares the client’s address or name information to each of that service’s filters—in order—using these criteria:

- The search stops at the first match. Because Allow filters are processed before Deny filters, Allow filters take precedence.
- Access is granted if the client information matches an Allow filter for that service.
- Access is denied if the client information matches a Deny filter for that service.
- If no match with any Allow or Deny filter occurs, access is granted—except in the case where there are Allow filters but no Deny filters, in which case lack of a match means that access is denied.

The filter syntax described here is flexible enough that you should be able to implement many different kinds of access-control policies in a simple and straightforward manner. You can use both Allow filters and Deny filters in any combination, even though you can probably implement most policies by using almost exclusively Allows or almost exclusively Denies.

The following sections describe filter syntax in detail and give usage examples. The section “[23.7.4 To Create Access Filters for Services](#)” on page [745](#) gives the procedure for creating access filters.

23.7.2 Filter Syntax

Filter statements contain both service information and client information. The service information can include the name of the service, names of hosts, and addresses of hosts. The client information can include host names, host addresses, and user names. Both the server and client information can include wildcard names or patterns.

The very simplest form of a filter is:

service: hostSpec

where *service* is the name of the service (such as `smtp`, `pop`, `imap`, or `http`) and *hostSpec* is the host name, IP address, or wildcard name or pattern that represents the client requesting access. When a filter is processed, if the client seeking access matches *client*, access is either allowed or denied (depending on which type of filter this is) to the service specified by *service*. Here are some examples:

```
imap: roberts.newyork.siroe.com  
pop: ALL  
http: ALL
```

If these are Allow filters, the first one grants the host `roberts.newyork.siroe.com` access to the IMAP service, and the second and third grant all clients access to the POP and HTTP services, respectively. If they are Deny filters, they deny those clients access to those services. (For descriptions of wildcard names such as `ALL`, see “[23.7.2.1 Wildcard Names](#)” on page [740](#).)

Either the server or the client information in a filter can be somewhat more complex than this, in which case the filter has the more general form of:

serviceSpec: clientSpec

where *serviceSpec* can be either *service* or *service@hostSpec*, and *clientSpec* can be either *hostSpec* or *user@hostSpec*. *user* is the user name (or a wildcard name) associated with the client host seeking access. Here are two examples:

```
pop@mailServer1.siroe.com: ALL  
imap: srashad@xyz.europe.siroe.com
```

If these are Deny filters, the first filter denies all clients access to the SMTP service on the host `mailServer1.siroe.com`. The second filter denies the user `srashad` at the host `xyz.europe.siroe.com` access to the IMAP service. (For more information on when to use these expanded server and client specifications, see “[23.7.2.4 Server-Host Specification](#)” on page [742](#) and “[23.7.2.5 Client User-Name Specification](#)” on page [742](#).)

Finally, at its most general, a filter has the form:

serviceList: clientList

where *serviceList* consists of one or more *serviceSpec* entries, and *clientList* consists of one or more *clientSpec* entries. Individual entries within *serviceList* and *clientList* are separated by blanks and/or commas.

In this case, when a filter is processed, if the client seeking access matches any of the *clientSpec* entries in *clientList*, then access is either allowed or denied (depending on which type of filter this is) to all the services specified in *serviceList*. Here is an example:

```
pop, imap, http: .europe.siroe.com .newyork.siroe.com
```

If this is an Allow filter, it grants access to POP, IMAP, and HTTP services to all clients in either of the domains `europe.siroe.com` and `newyork.siroe.com`. For information on using a leading dot or other pattern to specify domains or subnet, see “[23.7.2.2 Wildcard Patterns](#)” on [page 741](#).

You can also use the following syntax:

“+” or “-” *serviceList*:*\$*next_rule*

+ (allow filter) means the daemon list services are being granted to the client list.

- (deny filter) means the services are being denied to the client list.

* (wildcard filter) allow all clients to used these services.

\$ separates rules.

This example enables multiple services on all clients.

```
+imap, pop, http:*
```

This example shows multiple rules, but each rule is simplified to have only one service name and uses wildcards for the client list. (This is the most commonly used method of specifying access control in LDIF files.)

```
+imap:ALL$+pop:ALL$+http:ALL
```

An example of how to disallow all services for a user is:

```
-imap:*$-pop:*$-http:*
```

23.7.2.1

Wildcard Names

You can use the following wildcard names to represent service names, host names or addresses, or user names:

TABLE 23-3 Wildcard Names for Service Filters

Wildcard Name	Explanation
ALL, *	The universal wildcard. Matches all names.
LOCAL	Matches any local host (one whose name does not contain a dot character). However, if your installation uses only canonical names, even local host names will contain dots and thus will not match this wildcard.
UNKNOWN	<p>Matches any user whose name is unknown, or any host whose name or address is unknown.</p> <p>Use this wildcard name carefully:</p> <p>Host names may be unavailable due to temporary DNS server problems—in which case all filters that use UNKNOWN will match all client hosts.</p> <p>A network address is unavailable when the software cannot identify the type of network it is communicating with—in which case all filters that use UNKNOWN will match all client hosts on that network.</p>
KNOWN	<p>Matches any user whose name is known, or any host whose name and address are known.</p> <p>Use this wildcard name carefully:</p> <p>Host names may be unavailable due to temporary DNS server problems—in which case all filters that use KNOWN will fail for all client hosts.</p> <p>A network address is unavailable when the software cannot identify the type of network it is communicating with—in which case all filters that use KNOWN will fail for all client hosts on that network.</p>
DNSSPOOFER	Matches any host whose DNS name does not match its own IP address.

23.7.2.2 Wildcard Patterns

You can use the following patterns in service or client addresses:

- A string that begins with a dot character (.). A host name is matched if the last components of its name match the specified pattern. For example, the wildcard pattern `.siroe.com` matches all hosts in the domain `siroe.com`.
- A string that ends with a dot character (.). A host address is matched if its first numeric fields match the specified pattern. For example, the wildcard pattern `123.45.` matches the address of any host in the subnet `123.45.0.0`.
- A string of the form `n.n.n.n/m.m.m.m`. This wildcard pattern is interpreted as a *net/mask* pair. A host address is matched if *net* is equal to the bitwise AND of the address and *mask*. For example, the pattern `123.45.67.0/255.255.255.128` matches every address in the range `123.45.67.0` through `123.45.67.127`.

23.7.2.3 EXCEPT Operator

The access-control system supports a single operator. You can use the EXCEPT operator to create exceptions to matching names or patterns when you have multiple entries in either *serviceList* or *clientList*. For example, the expression:

list1 EXCEPT *list2*

means that anything that matches *list1* is matched, *unless* it also matches *list2*.

Here is an example:

ALL: ALL EXCEPT `isserver.siroe.com`

If this were a Deny filter, it would deny access to all services to all clients except those on the host machine `isserver.siroe.com`.

EXCEPT clauses can be nested. The expression:

list1 EXCEPT *list2* EXCEPT *list3*

is evaluated as if it were:

list1 EXCEPT (*list2* EXCEPT *list3*)

23.7.2.4

Server-Host Specification

You can further identify the specific service being requested in a filter by including server host name or address information in the *serviceSpec* entry. In that case the entry has the form:

`service@hostSpec`

You might want to use this feature when your Messaging Server host machine is set up for multiple internet addresses with different internet host names. If you are a service provider, you can use this facility to host multiple domains, with different access-control rules, on a single server instance.

23.7.2.5

Client User-Name Specification

For client host machines that support the `identd` service as described in RFC 1413, you can further identify the specific client requesting service by including the client's user name in the *clientSpec* entry in a filter. In that case the entry has the form:

`user@hostSpec`

where *user* is the user name as returned by the client's `identd` service (or a wildcard name).

Specifying client user names in a filter can be useful, but keep these caveats in mind:

- The `identd` service is not authentication; the client user name it returns cannot be trusted if the client system has been compromised. In general, do not use specific user names; use only the wildcard names `ALL`, `KNOWN`, or `UNKNOWN`.
- `identd` is not supported by most modern client machines and thus provides little added value in modern deployments. We are considering removal of `identd` support in a future version, so please inform Sun Java System if this feature is of value to your site.

- User-name lookups take time; performing lookups on all users may slow access by clients that do not support identd. Selective user-name lookups can alleviate this problem. For example, a rule like:

```
serviceList: @xyzcorp.com ALL@ALL
```

would match users in the domain xyzcorp.com without doing user-name lookups, but it would perform user-name lookups with all other systems.

The user-name lookup capability can in some cases help you guard against attack from unauthorized users on the client's host. It is possible in some TCP/IP implementations, for example, for intruders to use rsh (remote shell service) to impersonate trusted client hosts. If the client host supports the ident service, you can use user-name lookups to detect such attacks.

23.7.3 Filter Examples

The examples in this section show a variety of approaches to controlling access. In studying the examples, keep in mind that Allow filters are processed before Deny filters, the search terminates when a match is found, and access is granted when no match is found at all.

The examples listed here use host and domain names rather than IP addresses. Remember that you can include address and netmask information in filters, which can improve reliability in the case of name-service failure.

23.7.3.1 Mostly Denying

In this case, access is denied by default. Only explicitly authorized hosts are permitted access.

The default policy (no access) is implemented with a single, trivial deny file:

```
ALL: ALL
```

This filter denies all service to all clients that have not been explicitly granted access by an Allow filter. The Allow filters, then, might be something like these:

```
ALL: LOCAL @netgroup1
```

```
ALL: .siroe.com EXCEPT externalserver.siroe.com
```

The first rule permits access from all hosts in the local domain (that is, all hosts with no dot in their host name) and from members of the group netgroup1. The second rule uses a leading-dot wildcard pattern to permit access from all hosts in the siroe.com domain, with the exception of the host externalserver.siroe.com.

23.7.3.2 Mostly Allowing

In this case, access is granted by default. Only explicitly specified hosts are denied access.

The default policy (access granted) makes Allow filters unnecessary. The unwanted clients are listed explicitly in Deny filters such as these:

```
ALL: externalserver.siroe1.com, .siroe.asia.com  
ALL EXCEPT pop: contractor.siroe1.com, .siroe.com
```

The first filter denies all services to a particular host and to a specific domain. The second filter permits nothing but POP access from a particular host and from a specific domain.

23.7.3.3

Denying Access to Spoofed Domains

You can use the DNSSPOOFER wildcard name in a filter to detect host-name spoofing. When you specify DNSSPOOFER, the access-control system performs forward or reverse DNS lookups to verify that the client's presented host name matches its actual IP address. Here is an example for a Deny filter:

```
ALL: DNSSPOOFER
```

This filter denies all services to all remote hosts whose IP addresses don't match their DNS host names.

23.7.3.4

Controlling Access to Virtual Domains

If your messaging installation uses virtual domains, in which a single server instance is associated with multiple IP addresses and domain names, you can control access to each virtual domain through a combination of Allow and Deny filters. For example, you can use Allow filters like:

```
ALL@msgServer.siroe1.com: @.siroe1.com  
ALL@msgServer.siroe2.com: @.siroe2.com  
...
```

coupled with a Deny filter like:

```
ALL: ALL
```

Each Allow filter permits only hosts within domainN to connect to the service whose IP address corresponds to msgServer.siroeN.com. All other connections are denied.

23.7.3.5

Controlling IMAP Access While Permitting Access to Webmail

If you wish to allow users to access Webmail, but not access IMAP, create a filter like this:

```
+imap:access_server_host, access_server_host
```

This permits IMAP **only** from the access server hosts. You can set the filter at the IMAP server level using service imap.domainallowed, or at the domain/user level with LDAP attributes.

23.7.4 To Create Access Filters for Services

You can create Allow and Deny filters for the IMAP, POP, or HTTP services. You can also create them for SMTP services, but they have little value because they only apply to authenticated SMTP sessions. See [Chapter 18, “Mail Filtering and Access Control”](#)

▼ To Create Filters

- **Command Line.** You can also specify access and deny filters at the command line as follows:

To create or edit access filters for services:

```
configutil -o service.service.domainallowed -v filterwhere service is pop, imap, or http and  
filter follows the syntax rules described in “23.7.2 Filter Syntax” on page 739.
```

To create or edit deny filters for services:

```
configutil -o service.service.domainnotallowed -v filter
```

where *service* is pop, imap, or http and *filter* follows the syntax rules described in “[23.7.2 Filter Syntax](#)” on page 739. For a variety of examples, see “[23.7.3 Filter Examples](#)” on page 743

23.7.5 To Create Access Filters for HTTP Proxy Authentication

Any store administrator can proxy authenticate to any service. (For more information about store administrators, see “[20.4 Specifying Administrator Access to the Store](#)” on page 594) authenticate to the service if their client host is granted access via a proxy authentication access filter.

Proxy authentication allows other services, such as a portal site, to authenticate users and pass the authentication credentials to the HTTP login service. For example, assume a portal site offers several services, one of which is Messenger Express web-based email. By using the HTTP proxy authentication feature, end users need only authenticate once to the portal service; they need not authenticate again to access their email. The portal site must configure a login server that acts as the interface between the client and the service. To help configure the login server for Messenger Express authentication, Sun Java System offers an authentication SDK for Messenger Express.

This section describes how to create allow filters to permit HTTP proxy authentication by IP address. This section does not describe how to set up your login server or how to use the Messenger Express authentication SDK. For more information about setting up your login server for Messenger Express and using the authentication SDK, contact your Sun Java System representative.

▼ To Create Access Filters for HTTP Proxy Authentication

- **Command Line.** Specify access filters for proxy authentication to the HTTP service at the command line as follows:

```
configutil -o service.service.proxydomainallowed -v filter
```

where *filter* follows the syntax rules described in “[23.7.2 Filter Syntax](#)” on page 739.

23.8 Enabling POP Before SMTP

SMTP Authentication, or *SMTP Auth* (RFC 2554) is the preferred method of providing SMTP relay server security. SMTP Auth allows only authenticated users to send mail through the MTA. However, some legacy clients only provide support for *POP before SMTP*. If this is the case for your system, you may enable POP before SMTP as described below. If possible, however, encourage your users to upgrade their POP clients rather than using POP before SMTP. Once POP before SMTP is deployed at a site users will become dependent on clients which fail to follow Internet security standards, putting end users at greater risk of hacking and slowing your site with the unavoidable performance penalty because of the necessity of having to track and coordinate IP addresses of recent successful POP sessions.

The Messaging Server implementation of POP before SMTP is completely different from either SIMS or Netscape Messaging Server. POP before SMTP is supported by configuring a Messaging Multiplexor (MMP) to have both a POP and SMTP proxy. When an SMTP client connects to the SMTP proxy, the proxy will check an in-memory cache of recent POP authentications. If a POP authentication from the same client IP address is found, the SMTP proxy will inform the SMTP server that it should permit messages directed to both local and non-local recipients.

▼ To Install the SMTP Proxy

For guidelines on using the SMTP proxy see “[Using the MMP SMTP Proxy](#)” in *Sun Java Communications Suite 5 Deployment Planning Guide*.

- 1 **Install a Messaging Multiplexor (MMP).**

See the *Sun Java Communications Suite 5 Installation Guide* for instructions.

- 2 **Enable the SMTP proxy on the MMP.**

Add the string:

```
msg-svr-base/lib/SsmtpProxyAService@25|587
```

to the ServiceList option in the *msg-svr-base/config/AService.cfg* file. That option is one long line and can't contain line breaks.

Note – When the MMP is upgraded, four new files which correspond to the existing four configuration files for the MMP are created. The new files are:

`AService-def.cfg`, `ImapProxyAService-def.cfg`, `PopProxyAService-def.cfg`, and `SmtpProxyAService-def.cfg`

The four configuration files described in the documentation are not created or affected by the install process. When the MMP starts up, it will look for the normal configuration file (as currently documented). If it doesn't find the normal configuration file, it will attempt to copy the respective `*AService-def.cfg` file to the corresponding `*AService.cfg` file name.

3 Set the PROXY_PASSWORD option in the SMTP channel option file `tcp_local_option` at each SMTP relay server.

When the SMTP proxy connects to the SMTP server, it has to inform the SMTP server of the real client IP address and other connection information so that the SMTP server can correctly apply relay blocking and other security policy (including POP before SMTP authorization). This is a security sensitive operation and must be authenticated. The proxy password configured on both the MMP SMTP Proxy and the SMTP server assures that a third party cannot abuse the facility.

Example: `PROXY_PASSWORD=A_Password`

4 Make sure the IP address that the MMP uses to connect to the SMTP server is not treated as “internal” by the INTERNAL_IP mapping table.

See the “[18.6 To Add SMTP Relaying](#)” on page 557 of the [Chapter 18, “Mail Filtering and Access Control”](#) for information about the INTERNAL_IP mapping table.

5 Configure the SMTP proxy to Support POP before SMTP.

a. Edit the `msg-svr-base/config/SmtpProxyAService.cfg` configuration file.

The following SMTP proxy options operate identically to the same options for the IMAP and POP proxies (see [Chapter 7, “Configuring and Administering Multiplexor Services,”](#) these options in the Encryption (SSL) Option section in the “[Encryption \(SSL\) Option](#)” in [Sun Java System Messaging Server 6.3 Administration Reference](#).

`LdapURL`, `LogDir`, `LogLevel`, `BindDN`, `BindPass`, `Timeout`, `Banner`, `SSLEnable`, `SSLSecmodFile`, `SSLCertFile`, `SSLKeyFile`, `SSLKeyPasswdFile`, `SSLCipherSpecs`, `SSLCertNicknames`, `SSLCacheDir`, `SSLPorts`, `CertMapFile`, `CertmapDN`, `ConnLimits`, `TCPAccess`

Other MMP options not listed above (including the `BacksidePort` option) do not currently apply to the SMTP Proxy.

Add the following five options:

`SmtpRelays` is a space-separated list of SMTP relay server hostnames (with optional port) to use for round-robin relay. These relays must support the XPROXYEHLO extension. This option is mandatory with no default.

Example: `default:SmtpRelays manatee:485 gonzo mothra`

`SmtpProxyPassword` is a password used to authorize source channel changes on the SMTP relay servers. This option is mandatory with no default and must match the PROXY_PASSWORD option on the SMTP servers.

Example: `default:SmtpProxyPassword A_Password`

`EhloKeywords` option provides a list of EHLO extension keywords for the proxy to pass through to the client, in addition to the default set. The MMP will remove any unrecognized EHLO keywords from the EHLO list returned by an SMTP relay. `EhloKeywords` specifies additional EHLO keywords which should not be removed from the list. The default is empty, but the SMTP proxy will support the following keywords, so there is no need to list them in this option: 8BITMIME, PIPELINING, DSN, ENHANCEDSTATUSCODES, EXPN, HELP, XLOOP, ETRN, SIZE, STARTTLS, AUTH

The following is an example that might be used by a site which uses the rarely used “TURN” extension:

Example: `default:EhloKeywords TURN`

`PopBeforeSmtpKludgeChannel` option is set to the name of an MTA channel to use for POP before SMTP authorized connections. The default is empty and the typical setting for users who want to enable POP before SMTP is `tcp_intranet`. This option is not required for optimizing SSL performance (see “[23.5.4 How to Optimize SSL Performance Using the SMTP Proxy](#)” on page 734).

Example: `default:PopBeforeSmtpKludgeChannel tcp_intranet`

`ClientLookup` option defaults to no. If set to yes, a DNS reverse lookup on the client IP address will be performed unconditionally so the SMTP relay server doesn’t have to do that work. This option may be set on a per hosted domain basis.

Example: `default:ClientLookup yes`

- b. Set the `PreAuth` option and the `AuthServiceTTL` option in `PopProxyAService.cfg` configuration file. This option is not required for optimizing SSL performance. (See “[23.5.4 How to Optimize SSL Performance Using the SMTP Proxy](#)” on page 734)

These options specify how long in seconds a user is authorized to submit mail after a POP authentication. The typical setting is 900 to 1800 (15-30 minutes).

Example:

```
default:PreAuth yes  
default:AuthServiceTTL 900
```

- c. You may optionally specify how many seconds the MMP will wait for an SMTP Relay to respond before trying the next one in the list.

The default is 10 (seconds). If a connection to an SMTP Relay fails, the MMP will avoid trying that relay for a number of minutes equivalent to the failover time-out (so if the failover time-out is 10 seconds, and a relay fails, the MMP won't try that relay again for 10 minutes).

Example: default:FailoverTimeout 10

23.9 Configuring Client Access to SMTP Services

For information about configuring client access to SMTP services, see [Chapter 18, “Mail Filtering and Access Control”](#)

23.10 User/Group Directory Lookups Over SSL

It is possible to do user/group directory lookups over SSL for MTA, MMP and IMAP/POP/HTTP services. The prerequisite is that Messaging Server must be configured in SSL mode. Set the following configutil parameters to enable this feature:

`local.service.pab.ldapport to 636, local.ugldapport to 636, local.ugldapusessl to 1.`

Administering S/MIME for Communications Express Mail

Support for Secure/Multipurpose Internet Mail Extension (S/MIME) 3.1 is available on Sun Java System Communications Express Mail. Communications Express Mail users who are set up to use S/MIME can exchange signed or encrypted messages with other users of Communications Express Mail, Microsoft Outlook Express, and Mozilla mail systems.

Information about using S/MIME in Communications Express Mail is part of the online help. Information to administer S/MIME is explained in this chapter. This chapter consists of the following sections:

- “[24.1 What is S/MIME?](#)” on page 751
- “[24.2 Required Software and Hardware Components](#)” on page 753
- “[24.3 Requirements for Using S/MIME](#)” on page 754
- “[24.4 Getting Started After Installing Messaging Server](#)” on page 756
- “[24.5 Parameters of the smime.conf File](#)” on page 764
- “[24.6 Messaging Server Options](#)” on page 770
- “[24.7 Securing Internet Links With SSL](#)” on page 771
- “[24.8 Key Access Libraries for the Client Machines](#)” on page 772
- “[24.9 Verifying Private and Public Keys](#)” on page 774
- “[24.10 Granting Permission to Use S/MIME Features](#)” on page 781
- “[24.11 Managing Certificates](#)” on page 782
- “[24.12 Communications Express S/MIME End User Information](#)” on page 786

24.1 What is S/MIME?

Secure/Multipurpose Internet Mail Extensions (S/MIME) provides a consistent way for email users to send and receive secure MIME data, using digital signatures for authentication, message integrity and non-repudiation and encryption for privacy and data security. S/MIME version 3.1 (RFC 3851) is supported.

Several email clients support the S/MIME specification, including Microsoft Outlook Express and Mozilla mail.

You can deploy a secure mail solution by using Messaging Server and S/MIME. Communications Express webmail users who are set up to use S/MIME can exchange signed or encrypted messages with other users of Communications Express, Microsoft Outlook Express, and Mozilla mail systems. A messaging proxy can provide an additional layer of security at the firewall to further protect information assets within Messaging Server.

The Communications Express webmail client supports S/MIME with these features:

- Create a digital signature for an outgoing mail message to assure the message's recipient that the message was not tampered with and is from the person who sent it
- Encrypt an outgoing mail message to prevent anyone from viewing, changing or otherwise using the message's content before the message arrives in the recipient's mailbox
- Verify the digital signature of an incoming signed message with a process involving a certificate revocation list (CRL)
- Automatically decrypt an incoming encrypted message so the recipient can read the message's contents
- Exchange signed or encrypted messages with other users of an S/MIME compliant client such as Communications Express Mail and Mozilla mail systems

The remainder of this chapter describes how to configure Messaging Server and Communications Express for S/MIME. Note that you do not have to exclusively use Communications Express to be able to use S/MIME with Messaging Server.

24.1.1 Concepts You Need to Know

To properly administer S/MIME, you need to be familiar with the following concepts:

- Basic administrative procedures for your platform
- Structure and use of a lightweight directory access protocol (LDAP) directory
- Addition or modification of entries in an LDAP directory
- Configuration process for the Sun Java System Directory Server
- Concepts and purpose of the following:
 - Secure Socket Layer (SSL) for a secured communications line
 - Digitally signed email messages
 - Encrypted email messages
 - Local key store of a browser
 - Smart cards and the software and hardware to use them
 - Private-public key pairs and their certificates
 - Certificate authorities (CA)
 - Verifying keys and their certificates
 - Certificate revocation list (CRL). (See “[24.9.2 When is a Certificate Checked Against a CRL?](#)” on page [776](#))

24.2 Required Software and Hardware Components

The required hardware and software for using Communications Express Mail with S/MIME is described in this section. Ensure that you install all the correct versions of the software on the server and client machines before attempting to configure for S/MIME.

[Table 24–1](#) lists the required software and hardware for the client machine where Communications Express Mail is accessed.

TABLE 24–1 Required Hardware and Software for Client Machine

Component	Description
Operating system	<ul style="list-style-type: none"> ■ Microsoft Windows 98, 2000, or XP
Browser	<ul style="list-style-type: none"> ■ Microsoft Internet Explorer, Version 6 SP2 on Windows ■ Microsoft Internet Explorer, Version 6 SP1 (with current patches as of December 1, 2004) on Windows 2000 and Windows 98
Sun software	Java Runtime Environment Version 1.4.2_03 or later including Java 6, but not 1.5.1, nor 1.5.4 through 1.5.12. (That is, to use Version 1.5.x, a minimum of 1.5.2 is required, but do not use 1.5.4 through 1.5.12 due to incompatibilities with Smart Cards.)
Private-public keys with certificates	<p>One or more private-public key pair with certificates. Certificates are required and they must be in standard X.509 v3 format. Obtain keys and certificates from a CA for each Communications Express Mail user who will use the S/MIME features. The keys and their certificates are stored on the client machine or on a smart card. The public keys and certificates are also stored in an LDAP directory that can be accessed by Directory Server.</p> <p>A certificate revocation list (CRL), maintained by the CA, must be part of your system if you want key certificates checked against it to further ensure that the keys are valid. See “24.9.2 When is a Certificate Checked Against a CRL?” on page 776</p>
Smart card software (only required when keys and certificates are stored on smart cards)	<ul style="list-style-type: none"> ■ ActivCard Gold (now renamed ActiveIdentity), Version 2.1 or 3.0, or ■ NetSign, Version 3.1
Smart card reader	Any model of smart card reading device supported by the client machine and smart card software.

[Table 24–2](#) lists the required Sun Microsystems software for the server machine.

TABLE 24–2 Required Software for Server Machine

Sun Component	Description
Mail server	Sun Java System Messaging Server 6.5 Release or later on Solaris, Version 8 or 9, and Sun SPARC machine
LDAP server	Sun Java System Directory Server 5.2004Q2

TABLE 24–2 Required Software for Server Machine *(Continued)*

Sun Component	Description
Java	Java 2 Runtime Environment, Standard Edition, Version 1.4.2 and later including Java 6, but not 1.5.1, nor 1.5.4 through 1.5.12. (That is, to use Version 1.5.x, a minimum of 1.5.2 is required, but do not use 1.5.4 through 1.5.12 due to incompatibilities with Smart Cards.)
Access Manager	(If deployment is in Schema 2) - Sun Java System Access Manager 6 2005Q1 and Communications Express - Sun Java System Communications Express 6 2005Q1 or later.

24.3 Requirements for Using S/MIME

The signature and encryption features are not immediately available to Communications Express Mail users after you install Messaging Server. Before a user can take advantage of S/MIME, the requirements described in this section must be met.

24.3.1 Private and Public Keys

At least one private and public key pair, including a certificate in standard X.509 v3 format, must be issued to each Communications Express Mail user who will use S/MIME. The certificate, used in a verification process, assures other mail users that the keys really belong to the person who uses them. A user can have more than one key pair and associated certificate.

Keys and their certificates are issued from within your organization or purchased from a third-party vendor. Regardless of how the keys and certificates are issued, the issuing organization is referred to as a certificate authority (CA).

Key pairs and their certificates are stored in two ways:

- On a smart card

These cards are similar to commercial credit cards and should be used and safeguarded by the mail user as they do their own credit cards. Smart cards require special card readers attached to the mail user's computer (client machine) to read the private key information. See “[24.3.2 Keys Stored on Smart Cards](#)” on page 755 for more information.

- In a local key store on the mail user's computer (client machine)

A mail user's browser provides the key store. The browser also provides commands to download a key pair and certificate to the key store. See “[24.3.3 Keys Stored on the Client Machine](#)” on page 755 for more information.

24.3.2 Keys Stored on Smart Cards

If the private-public key pair, with its certificate, is stored on a smart card, a card reader must be properly attached to the mail user's computer. The card reading device also requires software; the device and its software are supplied by the vendor from whom you purchase this equipment.

There are actually two parts to a system with card reading capabilities. One part is the hardware card reader and its driver. The second part is the actual card, which is usually provided by a different vendor and requires drivers for reading the cards. Not all cards are supported. Refer to the [Table 24-1](#) to see a list of the supported SmartCards (ActiveCard, now renamed ActiveIdentity, and NetSign).

When properly installed, a mail user inserts their smart card into the reading device when they want to create a digital signature for an outgoing message. After verification of their smart card password, the private key is accessible by Communications Express Mail to sign the message. See [“24.2 Required Software and Hardware Components” on page 753](#) for information on supported smart cards and reading devices.

Libraries from the vendor of the smart card are required on the user's computer. See [“24.8 Key Access Libraries for the Client Machines” on page 772](#) for more information.

24.3.3 Keys Stored on the Client Machine

If key pairs and certificates are not stored on smart cards, they must be kept in a local key store on the mail user's computer (client machine). Their browser provides the key store and also has commands to download a key pair and certificate to the key store. The key store may be password-protected; this depends on the browser.

Libraries from the vendor of the browser are required on the user's computer to support a local key store. See [“24.8 Key Access Libraries for the Client Machines” on page 772](#) for more information.

24.3.4 Publish Public Keys in LDAP Directory

All public keys and certificates must also be stored to an LDAP directory, accessible by the Sun Java System Directory Server. This is referred to as publishing the public keys so they are available to other mail users who are creating S/MIME messages.

Public keys of the sender and receiver are used in the encrypting-decrypting process of an encrypted message. Public key certificates are used to validate private keys that were used for digital signatures.

See [“24.11 Managing Certificates” on page 782](#) for more information to use `ldapmodify` to publish the public keys and certificates.

24.3.5 Give Mail Users Permission to Use S/MIME

To create a signed or encrypted message, a valid Communications Express Mail user must have permission to do so. This involves using the `mailAllowedServiceAccess` or `mailDomainAllowedServiceAccess` LDAP attributes for a user's LDAP entry. These attributes can be used to include or exclude mail users from S/MIME on an individual or domain basis.

See “[24.10 Granting Permission to Use S/MIME Features](#)” on page 781 for more information.

24.3.6 Multi-language Support

A Communications Express Mail user who only uses English for their mail messages might not be able to read an S/MIME message which contains non-Latin language characters, such as Chinese. One reason for this situation is that the Java 2 Runtime Environment (JRE) installed on the user's machine does not have the `charsets.jar` file in the `/lib` directory.

The `charsets.jar` file is not installed if the English version of JRE was downloaded using the default JRE installation process. However, `charsets.jar` is installed for all other language choices of a default installation.

To ensure that the `charsets.jar` file is installed in the `/lib` directory, alert your users to use the custom installation to install the English version of JRE. During the installation process, the user must select the “Support for Additional Languages” option.

24.4 Getting Started After Installing Messaging Server

This section explains what the S/MIME applet is and provides a basic configuration procedure to set up S/MIME for Communications Express Mail. The configuration process involves setting parameters for the S/MIME applet and options for Messaging Server.

24.4.1 The S/MIME Applet

The process of signing a message, encrypting a message, or decrypting a message, along with the various procedures to verify private and public keys, are handled by a special applet, referred to as the S/MIME applet. The configuration of the S/MIME features is done with parameters in the `smime.conf` file and options of Messaging Server. [Figure 24–1](#) shows the S/MIME Applet in relation to other system components.

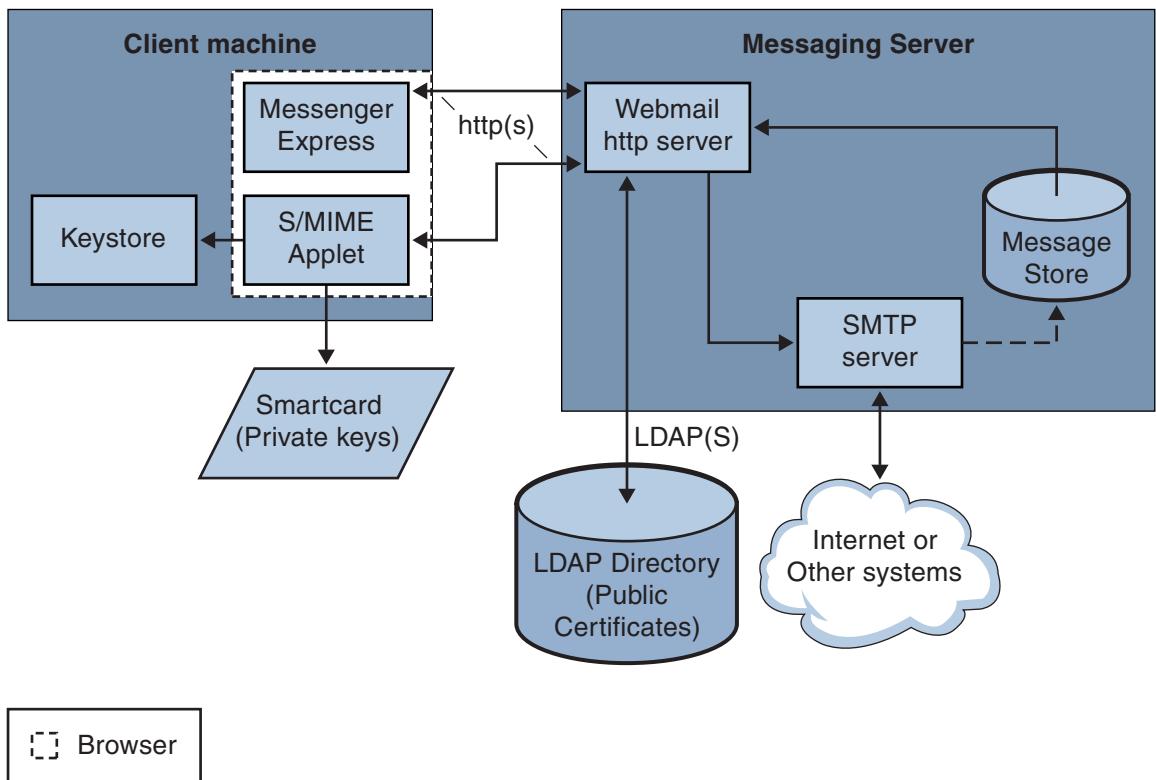


FIGURE 24-1 S/MIME Applet

24.4.1.1 Logging In for the First Time

When a Communications Express Mail user who has permission to use S/MIME logs in to the Messaging Server for the first time, a series of special prompts displays about the S/MIME applet. After answering the prompts with Yes or Always, the S/MIME applet is downloaded to their computer. The applet remains on their machine until they log out of Communications Express Mail.

Refer to “[24.11 Managing Certificates](#)” on page [782](#) for more information.

24.4.1.2 Downloading the S/MIME Applet

The S/MIME applet is downloaded each time a user logs in to Communications Express Mail unless caching is enabled for the Java 2 Runtime Environment (JRE) on the user’s machine. When caching is enabled, a copy of the S/MIME applet is saved on the user’s machine after the initial download which prevents downloading the applet every time the user logs in.

Caching can improve performance so you might direct your users to do the following steps to enable caching for Java 2 Runtime Environment, Version 1.4.x:

▼ To Enable Caching for Java 2 Runtime Environment, Version 1.4

- 1 Navigate to the Windows Control Panel.
- 2 Double click the Java Plug-in icon (Java 2 Runtime Environment).
- 3 Click the Cache tab.
- 4 Check the Enable Caching checkbox.
- 5 Click Apply.

After downloading, a user is not aware of the S/MIME applet. It appears that signing, encrypting, or decrypting a message is done by Communications Express Mail. Unless an error message pops up, the user also is unaware of the processes to verify a private or public key. Refer to “[24.9 Verifying Private and Public Keys](#)” on page [774](#) for more information.

24.4.2 A Basic S/MIME Configuration

The configuration file for S/MIME, `smime.conf`, contains descriptive comments and an example of each S/MIME parameter. The `smime.conf` file is included with Messaging Server, located in the directory `msg-svr-base/config/`, where `msg-svr-base` is the directory where Messaging Server is installed.

The following procedure contains the minimum required steps to configure the S/MIME features:

▼ To Configure the S/MIME

- 1 Verify that the basic features of Communications Express Mail are working after you install Messaging Server.
- 2 If you haven’t already, create or obtain private-public key pairs, with certificates in standard X.509 v3 format, for all your mail users who have permission to use the S/MIME features.
- 3 If smart cards are used for keys and certificates:
 - a. Distribute the smart cards to your mail users.
 - b. Ensure that the smart card reading devices and software are properly installed on each client machine where Communications Express Mail is accessed.
- 4 If local key stores of the browsers are used to store keys and certificates, instruct your mail users how to download their key pairs and certificate to the local key store.

- 5 Ensure that the correct libraries are on the client machines to support smart cards or local key stores. See “[24.8 Key Access Libraries for the Client Machines](#)” on page [772](#)
- 6 Set up your LDAP directory to support S/MIME:
 - a. Store all certificates for the CAs in the LDAP directory, accessible by Directory Server, under the distinguished name for certificate authorities. The LDAP attribute for these certificates is `cacertificate;binary`. Write down the directory information where you store them. You'll need this information for a later step.
See `trustedurl` in [Table 24–3](#) for an example of specifying LDAP directory information and “[24.11 Managing Certificates](#)” on page [782](#) for information to search an LDAP directory.
 - b. Store the public keys and certificates in the LDAP directory accessible by Directory Server. The LDAP attribute for public keys and certificates is `usercertificate;binary`. Write down the directory information where you store them. You'll need this information for a later step.
See `certurl` in [Table 24–3](#) for an example of specifying LDAP directory information and “[24.11 Managing Certificates](#)” on page [782](#) for information to search an LDAP directory.
 - c. Ensure that all users who send or receive S/MIME messages are given permission to use S/MIME with an LDAP filter in their user entries. A filter is defined with the `mailAllowedServiceAccess` or `mailDomainAllowedServiceAccess` LDAP attributes.
Note: By default, if you do not use `mailAllowedServiceAccess` or `mailDomainAllowedServiceAccess`, all services including `smime`, are allowed. If you explicitly specify services with these attributes, then the services `http` and `smtp`, as well as `smime`, must be specified to give mail users permission to use the S/MIME features.
See “[24.10 Granting Permission to Use S/MIME Features](#)” on page [781](#) for more information.
- 7 Edit the `smime.conf` file with any available text editor. See comments at the beginning of the file for parameter syntax.
All text and example parameters in `smime.conf` are preceded with a comment character (#). You can add the parameters you need to `smime.conf` or copy a parameter example to another part of the file and change its value. If you copy and edit an example, be sure to remove the # character at the beginning of its line.
Add these parameters to the file, each on its own line:
 - a. `trustedurl` (see [Table 24–3](#))-- set to the LDAP directory information to locate the certificates of the CAs. Use the information you saved from Step a.
 - b. `certurl` ([Table 24–3](#))-- set to the LDAP directory information to locate the public keys and certificates. Use the information you saved from Step b.

- c. `usersertfilter` (see [Table 24–3](#)) -- set to the value of the example in the `smime.conf` file. The example value is almost always the filter you want. Copy the example and delete the # character at the beginning of the line.

This parameter specifies a filter definition for the primary, alternate, and equivalent email addresses of a Communications Express Mail user to ensure that all of a user's private-public key pairs are found when the key pairs are assigned to different mail addresses.

- d. `sslrootcacertsurl` (see [Table 24–3](#))-- if you are using SSL for the communications link between the S/MIME applet and Messaging Server, set `sslrootcacertsurl` with the LDAP directory information to locate the certificates of CAs that are used to verify the Messaging Server's SSL certificates. See "[24.7 Securing Internet Links With SSL](#)" on page [771](#) for more information.
`checkoverssl` (see [Table 24–3](#))-- set to 0 if you are not using SSL for the communications link between the S/MIME applet and Messaging Server.
- e. `crlenable` (see [Table 24–3](#))-- set to 0 to disable CRL checking for now because doing CRL checking might require adding other parameters to the `smime.conf` file.
- f. `logindn` and `loginpw` ([Table 24–3](#))-- if the LDAP directory that contains the public keys and CA certificates requires authentication to access it, set these parameters to the distinguished name and password of the LDAP entry that has read permission.

Note: The values of `logindn` and `loginpw` are used whenever the LDAP directory is accessed with the LDAP information specified by the `crlmappingurl`, `sslrootcacertsurl`, or `trustedurl` parameters. See "[24.5 Parameters of the smime.conf File](#)" on page [764](#) and "[24.4.3 Accessing LDAP for Public Keys, CA certificates and CRLs Using Credentials](#)" on page [762](#) for more information.

Do not set `logindn` and `loginpw` if authentication is not required to access the LDAP directory.

8 Set the Messaging Server options with configutil:

- a. `local.webmail.smime.enable`-- set to 1.
- b. `local.webmail.cert.enable`-- set to 1 if you want to verify certificates against a CRL. See "[24.6 Messaging Server Options](#)" on page [770](#) for more information.

9 Communications Express Mail is now configured for the S/MIME features. Verify that the S/MIME features are working with the following steps:

- a. Restart the Messaging Server.
- b. Check the Messaging Server log file, `msg-svr-base /log/http`, for diagnostic messages relating to S/MIME.

- c. If any problems were detected for S/MIME, the diagnostic messages help you determine how to correct the problem with the configuration parameters.
- d. Correct the necessary configuration parameters.
- e. Repeat Steps a. through d. until there are no more diagnostic messages for S/MIME in the Messaging Server's log file.
- f. Check that the S/MIME features are working with the following steps:
 - i. Log in to Messaging Server from a client machine. Answer the special prompts for the S/MIME applet with Yes or Always. See "[24.11 Managing Certificates](#)" on page [782](#)
 - ii. Compose a short message, addressed to yourself.
 - iii. Encrypt your message by checking the Encrypt checkbox at the bottom of the Compose window if it is not already checked.
 - iv. Click Send to send the encrypted message to yourself. This should exercise most of the mechanisms for keys and certificates.
 - v. If you find problems with the encrypted message, the most likely causes are the values you used for LDAP directory information in the `smime.conf` file and/or the way keys and certificates are stored in the LDAP directory. Check the Messaging Server log for more diagnostic messages.

The remaining S/MIME parameters, summarized in the table below, provide many options you might want to use to further configure your S/MIME environment. See "[24.5 Parameters of the smime.conf File](#)" on page [764](#) for more information about the parameters.

Required Parameters for S/MIME	Parameters for Smart Cards and Local Key Stores	Parameters for CRL Checking	Parameters for Initial Settings and Secured Links
certurl*	platformwin	checkoverssl	alwaysencrypt
logindn		crlaccessfail	alwaysign
loginpw		crldir	sslrootcacertsurl
trustedurl*		crlenable	
usercertfilter*		crlmappingurl	
		crlurllogindn	
		crlurlloginpw	

Required Parameters for S/MIME	Parameters for Smart Cards and Local Key Stores	Parameters for CRL Checking	Parameters for Initial Settings and Secured Links
		crlusepastnextupdate	
		readsigncert	
		revocationunknown	
		sendencryptcert	
		sendencryptcertrevoked	
		readsigncert	
		sendsigncertrevoked	
		timestampdelta	

* You must specify a value for these parameters because they have no default value.

24.4.3 Accessing LDAP for Public Keys, CA certificates and CRLs Using Credentials

Public keys, CA certificates, and CRLs required for S/MIME may be stored in an LDAP directory (see previous section). The keys, certificates, and CRLs may be accessible from a single URL or multiple URLs in LDAP. For example, CRLs may be stored in one URL and public keys and certificates may be stored in another. Messaging Server allows you to specify which URL contains the desired CRL or certificate information, as well as the DN and password of the entry that has access to these URLs. These DN/password credentials are optional; if none are specified, LDAP access first tries the HTTP server credentials, and if that fails, it tries accessing it as anonymous.

Two pairs of `smime.conf` credential parameters may be set to access the desired URLs: `logindn` and `loginpw`, and `crlurllogindn` and `crlurlloginpw`.

`logindn` and `loginpw` are the credentials used for all URLs in `smime.conf`. They specify the DN and password of the LDAP entry that has read permission for the public keys, their certificates, and the CA certificates as specified by the `certurl` and `trustedurl` parameters.

`crlurllogindn` and `crlurlloginpw` specifies the DN and password of the LDAP entry that has read permission for the resulting URL from the mapping table (see “[24.9.3 Accessing a CRL](#) on page [777](#) for more information). If these credentials are NOT accepted, LDAP access is denied and no retry with other credentials is attempted. Either both parameters must be specified, or both must be empty. These parameters do not apply to the URLs that come directly from the certificate.

24.4.3.1 Setting Passwords for Specific URLs

Messaging Server allows you to specifically define the DN/password pairs for accessing the following `smime.conf` URLs: `certUrl`, `trustedUrl`, `crlmappingUrl`, `sslrootcacertsUrl`.

The syntax is as follows:

`url_type URL[|URL_DN | URL_password]`

Example:

```
trustedurl==ldap://mail.siroe.com:389/cn=Directory Manager, ou=people,
o=siroe.com,o=ugroot?cacertificate?sub?(objectclass=certificationauthority) |
cn=Directory manager | boomshakalaka
```

24.4.3.2 Summary of Using LDAP credentials

This section summarizes the use of LDAP credentials.

- All LDAP credentials are optional; if none are specified, LDAP access first tries the HTTP server credentials, and if that fails, tries anonymous.

Two pairs of `smime.conf` parameters are used as credentials for the two sets of URLs that may be specified:

`logindn` & `loginpw` - all URLs in `smime.conf`

`crlurllogindn` & `crlurlloginpw` - all URLs from mapping table

These are known as the default LDAP credential pair.

- Any URL specified in `smime.conf` or via mapping CRL URLs can have an optional local LDAP credential pair specified.
- Credentials are checked in order in which each is specified:
 - 1) Local LDAP credential pair - if specified, only one tried
 - 2) Default LDAP Credential Pair - if specified, and no Local LDAP credential pair, only one tried
 - 3) Server - if neither Local LDAP credential pair nor default LDAP credential pair specified, first tried
 - 4) anonymous - last tried only if server fails or none specified
- If a URL has a Local LDAP credential pair specified, it is used first; if the access fails, access is denied.
- If a URL has no Local LDAP credential pair specified, the corresponding default LDAP credential pair is used; if access fails, then access is denied.

24.5 Parameters of the smime.conf File

The `smime.conf` file is included with the Messaging Server, located in the directory `msg-svr-base/config/`, where `msg-svr-base` is the directory where Messaging Server is installed. All text and parameter examples in the file are preceded with a comment character (#).

You can add parameters with your values to the `smime.conf` file or you can edit the parameter examples. If using an example, copy the example to another part of the file, edit the parameter's value, and remove the # character at the beginning of the line.

Edit `smime.conf` with any available text editor after you install Messaging Server. The parameters, described in [Table 24–3](#), are not case sensitive and unless otherwise stated, are not required to be set.

TABLE 24–3 S/MIME Configuration Parameters in `smime.conf` File

Parameter	Purpose
<code>alwaysencrypt</code>	<p>Controls the initial setting for whether all outgoing messages are automatically encrypted for all Communications Express Mail users with permission to use S/MIME. Each Communications Express Mail user can override this parameter's value for their messages by using the checkboxes described in Table 24–5.</p> <p>Choose one of these values:</p> <ul style="list-style-type: none"> 0 - do not encrypt messages. The encryption checkboxes within Communications Express Mail are displayed as unchecked. This is the default. 1 - always encrypt messages. The encryption checkboxes within Communications Express Mail are displayed as checked. <p>Example:</p> <pre>alwaysencrypt==1</pre>
<code>alwaysign</code>	<p>Controls the initial setting for whether all outgoing messages are automatically signed for all Communications Express Mail users with permission to use S/MIME. Each Communications Express Mail user can override this parameter's value for their messages by using the checkboxes described in Table 24–5.</p> <p>Choose one of these values:</p> <ul style="list-style-type: none"> 0 - do not sign messages. The signature checkboxes within Communications Express Mail are displayed as unchecked. This is the default. 1 - always sign messages. The signature checkboxes within Communications Express Mail are displayed as checked. <p>Example:</p> <pre>alwaysign==1</pre>

TABLE 24-3 S/MIME Configuration Parameters in smime.conf File (Continued)

Parameter	Purpose
certurl	<p>Specifies the LDAP directory information to locate the public keys and certificates of Communications Express Mail users (the LDAP attribute for public keys is <code>usercertificate;binary</code>). See “24.11 Managing Certificates” on page 782 for more information about certificates.</p> <p>This parameter must point to the highest node in the user/group of the LDAP directory information tree (DIT) that includes all users that are being served by the Messaging Server. This is particularly important for sites with more than one domain; the distinguished name must be the root distinguished name of the user/group tree instead of the subtree that contains users for a single domain.</p> <p>This is a required parameter that you must set.</p> <p>Example:</p> <pre>certurl==ldap://mail.siroe.com:389/ou=people,o=siroe.com,o=ugroot</pre>
checkoverssl	<p>Controls whether an SSL communications link is used when checking a key’s certificate against a CRL. See “24.7 Securing Internet Links With SSL” on page 771 for more information.</p> <p>Choose one of these values:</p> <ul style="list-style-type: none"> 0 - do not use an SSL communications link. 1 - use an SSL communications link. This is the default. <p>A problem can occur when a proxy server is used with CRL checking in effect. See “24.9.4 Proxy Server and CRL Checking” on page 778</p>
crlaccessfail	<p>Specifies how long to wait before the Messaging Server attempts to access a CRL after it has failed to do so after multiple attempts. This parameter has no default values.</p> <p>Syntax:</p> <pre>crlaccessfail==number_of_failures:time_period_for_failures:wait_time_before_retry</pre> <p>where:</p> <p><i>number_of_failures</i> is the number of times that the Messaging Server can fail to access a CRL during the time interval specified by <i>time_period_for_failures</i>. The value must be greater than zero.</p> <p><i>time_period_for_failures</i> is the number of seconds over which the Messaging Server counts the failed attempts to access a CRL. The value must be greater than zero.</p> <p><i>wait_time_before_retry</i> is the number of seconds that the Messaging Server waits, once it detects the limit on failed attempts over the specified time interval, before trying to access the CRL again. The value must be greater than zero.</p> <p>Example:</p> <pre>crlaccessfail==10:60:300</pre> <p>In this example, Messaging Server fails 10 times within a minute to access the CRL. It then waits 5 minutes before attempting to access the CRL again. See “24.9.7 Trouble Accessing a CRL” on page 780</p>

TABLE 24–3 S/MIME Configuration Parameters in smime.conf File (Continued)

Parameter	Purpose
crldir	Specifies the directory information where the Messaging Server downloads a CRL to disk. The default is <i>msg-svr-base</i> /data/store/mboxlist, where <i>msg-svr-base</i> is the directory where Messaging Server is installed. See “ 24.9.5 Using a Stale CRL ” on page 778 for more information.
crlenable	<p>Controls whether a certificate is checked against a CRL. If there is a match, the certificate is considered revoked. The values of the <i>send*revoked</i> parameters in the <i>smime.conf</i> file determine whether a key with a revoked certificate is rejected or used by Communications Express Mail. See “24.9 Verifying Private and Public Keys” on page 774 for more information.</p> <p>Choose one of these values:</p> <ul style="list-style-type: none"> 0- each certificate is not checked against a CRL. 1- each certificate is checked against a CRL. This is the default. Ensure that the <i>local.webmail.cert.enable</i> option of the Messaging Server is set to 1, otherwise CRL checking is not done even if <i>crlenable</i> is set to 1.
crlmappingurl	<p>Specifies the LDAP directory information to locate the CRL mapping definitions. This parameter is only required when you have mapping definitions. See “24.9.3 Accessing a CRL” on page 777 optionally add the DN and password that has access to the URL.</p> <p>Syntax:</p> <pre>crlmappingurl URL[URL_DN URL_password]</pre> <p>Example:</p> <pre>crlmappingurl==ldap://mail.siroe.com:389/cn=XYZ Messaging, ou=people, o=mail.siroe.com,o=isp?msgCRLMappingRecord?sub?(objectclass=msgCRLMappingTable) cn=Directory Manager pAsswOrD</pre>
crlurllogindn	<p>Specifies the distinguished name of the LDAP entry that has read permission for the CRL mapping definitions (not if the entry is directly from the certificate, see “Accessing a CRL” on page 904 for more information).</p> <p>If values for <i>crllogindn</i> and <i>crlloginpw</i> are not specified, the Messaging Server uses the log in values for the HTTP server to gain entry to the LDAP directory. If that fails, Messaging Server attempts to access the LDAP directory anonymously.</p> <p>Example:</p> <pre>crllogindn==cn=Directory Manager</pre>
crlurlloginpw	<p>Specifies the password, in ASCII text, for the distinguished name of the <i>crllogindn</i> parameter.</p> <p>If values for <i>crllogindn</i> and <i>crlloginpw</i> are not specified, Messaging Server uses the log in values for the HTTP server to gain entry to the LDAP directory. If that fails, Messaging Server attempts to access the LDAP directory anonymously.</p> <p>Example:</p> <pre>crlloginpw==zippy</pre>

TABLE 24-3 S/MIME Configuration Parameters in smime.conf File *(Continued)*

Parameter	Purpose
crlusepastnextupdate	<p>Controls whether a CRL is used when the current date is past the date specified in the CRL's next-update field. See "24.9.5 Using a Stale CRL" on page 778 for more information.</p> <p>Choose one of these values:</p> <p>0 - do not use the stale CRL.</p> <p>1 - use the stale CRL. This is the default.</p>
logindn	<p>Specifies the distinguished name of the LDAP entry that has read permission for the public keys and their certificates, and the CA certificates located in the LDAP directory specified by the certurl and trustedurl parameters.</p> <p>If values for logindn and loginpw are not specified, the Messaging Server uses the log in values for the HTTP server to gain entry to the LDAP directory. If that fails, Messaging Server attempts to access the LDAP directory anonymously.</p> <p>Example:</p> <pre>logindn==cn=Directory Manager</pre>
loginpw	<p>Specifies the password, in ASCII text, for the distinguished name of the logindn parameter.</p> <p>If values for logindn and loginpw are not specified, Messaging Server uses the log in values for the HTTP server to gain entry to the LDAP directory. If that fails, Messaging Server attempts to access the LDAP directory anonymously.</p> <p>Example:</p> <pre>loginpw==SkyKing</pre>
platformwin	<p>Specifies one or more library names that are necessary when using smart cards or a local key store on a Windows platform. Change this parameter only if the default value does not work for your client machines. The default is:</p> <pre>platformwin==CAPI:library=capibridge.dll;</pre> <p>See "24.8 Key Access Libraries for the Client Machines" on page 772 for more information.</p>
readsigncert	<p>Controls whether a public key's certificate is checked against a CRL to verify an S/MIME digital signature when the message is read. (A private key is used to create a digital signature for a message but it cannot be checked against a CRL, so the certificate of the public key associated with the private key is checked against the CRL.) See "24.9 Verifying Private and Public Keys" on page 774</p> <p>Choose one of these values:</p> <p>0 - do not check the certificate against a CRL.</p> <p>1 - check the certificate against a CRL. This is the default.</p>

TABLE 24–3 S/MIME Configuration Parameters in smime.conf File *(Continued)*

Parameter	Purpose
revocationunknown	<p>Determines the action to take when an ambiguous status is returned when checking a certificate against a CRL. In this case, it is not certain whether the certificate is valid or has a revoked status. See “24.9 Verifying Private and Public Keys” on page 774 for more information.</p> <p>Choose one of these values:</p> <ul style="list-style-type: none"> ok - treat the certificate as valid. revoked - treat the certificate as revoked. This is the default.
sendencryptcert	<p>Controls whether the certificate of a public key that is used to encrypt an outgoing message is checked against a CRL before using it. See “24.9 Verifying Private and Public Keys” on page 774</p> <p>Choose one of these values:</p> <ul style="list-style-type: none"> 0 - do not check the certificate against a CRL. 1 - check the certificate against a CRL. This is the default.
sendencryptcertrevoked	<p>Determines the action to take if the certificate of a public key that is used to encrypt an outgoing message is revoked. See “24.9 Verifying Private and Public Keys” on page 774 for more information.</p> <p>Choose one of these values:</p> <ul style="list-style-type: none"> allow - use the public key. disallow - do not use the public key. This is the default.
sendsigncert	<p>Controls whether a public key’s certificate is checked against a CRL to determine if a private key can be used to create a digital signature for an outgoing message. (A private key is used for a digital signature but it cannot be checked against a CRL, so the certificate of the public key associated with the private key is checked against the CRL.) See “24.9 Verifying Private and Public Keys” on page 774 for more information.</p> <p>Choose one of these values:</p> <ul style="list-style-type: none"> 0 - do not check the certificate against a CRL. 1 - check the certificate against a CRL. This is the default.
sendsigncertrevoked	<p>Determines the action to take when it is determined that a private key has a revoked status. (A private key is used to create a digital signature for a message but it cannot be checked against a CRL, so the certificate of the public key associated with the private key is checked against the CRL. If the public key certificate is revoked, then its corresponding private key is also revoked.) See “24.9 Verifying Private and Public Keys” on page 774 for more information.</p> <p>Choose one of these values:</p> <ul style="list-style-type: none"> allow - use the private key with a revoked status. disallow - do not use the private key with a revoked status. This is the default.

TABLE 24-3 S/MIME Configuration Parameters in smime.conf File (Continued)

Parameter	Purpose
sslrootcacertsurl	<p>Specifies the distinguished name and the LDAP directory information to locate the certificates of valid CAs which are used to verify the Messaging Server's SSL certificates. This is a required parameter when SSL is enabled in the Messaging Server. See “24.7 Securing Internet Links With SSL” on page 771 for more information.</p> <p>If you have SSL certificates for a proxy server that receives all requests from client application, the CA certificates for those SSL certificates must also be located in the LDAP directory pointed to by this parameter.</p> <p>You can also optionally add the DN and password that has access to the URL.</p> <p>Syntax:</p> <pre>crlmappingurl URL[URL_DN URL_password]</pre> <p>Example:</p> <pre>sslrootcacertsurl==ldap://mail.siroe.com:389/cn=SSL Root CA Certs,ou=people,o=siroe.com,o=isp?cacertificate;binary?base? (objectclass=certificationauthority) cn=Directory Manager pAsSwOrD</pre>
timestampdelta	<p>Specifies a time interval, in seconds, that is used to determine whether a message's sent time or received time is used when checking a public key's certificate against a CRL.</p> <p>The parameter's default value of zero directs Communications Express Mail to always use the received time. See “24.9.6 Determining Which Message Time to Use” on page 779 for more information.</p> <p>Example:</p> <pre>timestampdelta==360</pre>
trustedurl	<p>Specifies the distinguished name and LDAP directory information to locate the certificates of valid CAs. This is a required parameter.</p> <p>You can also optionally add the DN and password that has access to the URL.</p> <p>Syntax:</p> <pre>crlmappingurl URL[URL_DN URL_password]</pre> <p>Example:</p> <pre>trustedurl==ldap://mail.siroe.com:389/cn=Directory Manager, ou=people, o=siroe.com, o=ugroot?cacertificate?sub? (objectclass=certificationauthority) cn=Directory Manager pAsSwOrD</pre>

TABLE 24-3 S/MIME Configuration Parameters in smime.conf File *(Continued)*

Parameter	Purpose
usercertfilter	<p>Specifies a filter definition for the primary, alternate, and equivalent email addresses of a Communications Express Mail user to ensure that all of a user's private-public key pairs are found when they are assigned to different mail addresses.</p> <p>This parameter is required and has no default values.</p>

24.6 Messaging Server Options

To set the three Messaging Server options that apply to S/MIME, do the following on the machine where the Messaging Server is installed.

▼ To Set Messaging Server Options that Apply to S/MIME

- 1 Log in as root. Then enter:

```
# cd msg-svr-base/sbin
```

where *msg-svr-base* is the directory where Messaging Server is installed.

- 2 Set the Messaging Server options, described in the following table, as desired for your system. Use the `configutil` utility to set them. Unless stated otherwise, an option is not required to be set.

Parameter	Purpose
local.webmail.cert.enable	<p>Controls whether the process that handles CRL checking should do CRL checking.</p> <p>0 - the process does not check a certificate against a CRL. This is the default.</p> <p>1 - the process checks a certificate against a CRL. When set to 1, ensure that the <code>crlenable</code> parameter in the <code>smime.conf</code> file is set to 1.</p>
local.webmail.cert.port	<p>Specifies a port number on the machine where the Messaging Server runs to use for CRL communication. This port is used locally for that machine only. The value must be greater than 1024. The default is 55443.</p> <p>This is a required option if the default port number is already in use.</p>

Parameter	Purpose
<code>local.webmail.smime.enable</code>	<p>Controls whether the S/MIME features are available to Communications Express Mail users. Choose one of these values:</p> <p>0 - the S/MIME features are unavailable for Communications Express Mail users even though the system is configured with the correct software and hardware components. This is the default.</p> <p>1 - the S/MIME features are available to Communications Express Mail users who have permission to use them.</p> <p>Example:</p> <pre>configutil -o local.webmail.smime.enable -v 1</pre>

24.7 Securing Internet Links With SSL

The Messaging Server supports the use of the Secure Socket Layer (SSL) for Internet links affecting Communications Express Mail, as summarized in the following table.

Link Between:	Description
Messaging Server and Communications Express Mail	<p>Securing this link with SSL requires administrative work for the Messaging Server. The Communications Express Mail user must use the HTTPS protocol, rather than HTTP, when entering the URL information for the Messaging Server in their browser.</p> <p>See “24.7.1 Securing the Link Between Messaging Server and Communications Express Mail” on page 771</p>
Messaging Server and S/MIME applet	<p>When checking public keys certificates against a CRL, the S/MIME applet must communicate directly with the Messaging Server. Securing this link with SSL requires administrative work for the Messaging Server in addition to setting <code>sslrootcacertsurl</code> and <code>checkoverssl</code> in the <code>smime.conf</code> file.</p> <p>See “24.7.2 Securing the Link Between the Messaging Server and S/MIME Applet” on page 772</p>

24.7.1 Securing the Link Between Messaging Server and Communications Express Mail

The Messaging Server supports the use of Secure Socket Layer (SSL) for the Internet link between it and Communications Express Mail. Once you have set up Messaging Server for SSL, configure Communications Express for SSL See [Sun Java System Communications Express 6.3 Administration Guide](#). A Communications Express Mail user specifies the Communications Express URL in their browser with the HTTPS protocol:

`HTTPS://hostname.domain:secured_port`

instead of the HTTP protocol (`HTTP://hostname.domain:unsecure_port`). When the Communications Express login window displays, the user sees a lock icon in a locked position at the bottom of their window to indicate they have a secure link.

See “[23.5 Configuring Encryption and Certificate-Based Authentication](#)” on page [720](#) for SSL configuration information for Messaging Server.

24.7.2 Securing the Link Between the Messaging Server and S/MIME Applet

When checking the certificate of a public key against a CRL, the S/MIME applet must communicate directly with the Messaging Server.

▼ To Secure the Communications Link with SSL

- 1 Do the administrative tasks to configure the Messaging Server for SSL. See “[23.5 Configuring Encryption and Certificate-Based Authentication](#)” on page [720](#).
- 2 Set the `sslrootcacertsurl` parameter in the `smime.conf` file to specify the information to locate the root SSL CA certificates. These CA certificates are used to verify the Messaging Server’s SSL certificates when the SSL link is established between the Messaging Server and the S/MIME applet.
- 3 Set the `checkoverssl` parameter in the `smime.conf` file to 1. This Messaging Server option determines whether SSL is used for the link between the Messaging Server and the S/MIME applet. Regardless of how a Communications Express Mail user specifies the URL for the Messenger Server (HTTP or HTTPS), the link between the Messaging Server and the S/MIME applet is secured with SSL when `checkoverssl` is set to 1.

Note – A proxy server can be used between the Messaging Server and client applications such as Communications Express Mail. See “[24.9.4 Proxy Server and CRL Checking](#)” on page [778](#) using a proxy server with and without a secured communications link.

24.8 Key Access Libraries for the Client Machines

Whether your mail users keep their private-public key pairs and certificates on a smart card or in a local key store of their browsers, key access libraries must be present on the client machines to support the storage methods.

The libraries are supplied by vendors of the smart cards and browsers. You must ensure that the correct libraries are on the client machines and specify the library name or names with the appropriate platform parameter in the `smime.conf` file. The parameters choices are:

- `platformwin` for Microsoft Windows running on a PC.

You can specify only the libraries you know are installed on the client machines or you can specify all the library names for a given platform and vendor if you are not sure what is installed. If the S/MIME applet does not find the library it needs among the names you specify, the S/MIME features do not work.

The syntax to specify one or more library filenames is:

`platform_parameter==vendor:library=library_name;...`

where:

platform_parameter is the parameter name for the platform of the client machine where Communications Express Mail is accessed. Choose one of these names: `platformwin`

vendor specifies the vendor of the smart card or browser. Choose one of these literals:

`cac` (for an ActivCard or NetSign smart card)

`capi` (for Internet Explorer with CAPI)

`mozilla` (for Mozilla with Network Security Services)

library_name specifies the library filename. See [Table 24–4](#) for the library name for your vendor and operating system.

TABLE 24–4 Special Libraries for the Client Machines

Smart Card or Browser Vendor	Operating System	Library Filename
	Windows	<code>acpkcs211.dll</code>
Internet Explorer with Cryptographic Application Programming Interface (CAPI)	Windows	<code>capibridge.dll</code>
	Windows	<code>softokn3.dll</code>
	Windows	<code>core32.dll</code>

24.8.1 Example

The following example specifies one smart card library and one Internet Explorer library, and one Mozilla library for a Windows platform:

```
platformwin==CAC:library=acpkcs211.dll;CAPI:library=capibridge.dll;
MOZILLA:library=softokn3.dll;
```

24.9 Verifying Private and Public Keys

Before Communications Express Mail uses a private or public key, it must pass the verification tests shown in [Figure 24–2](#). The remainder of this section describes the details of checking a public key's certificate against a CRL.

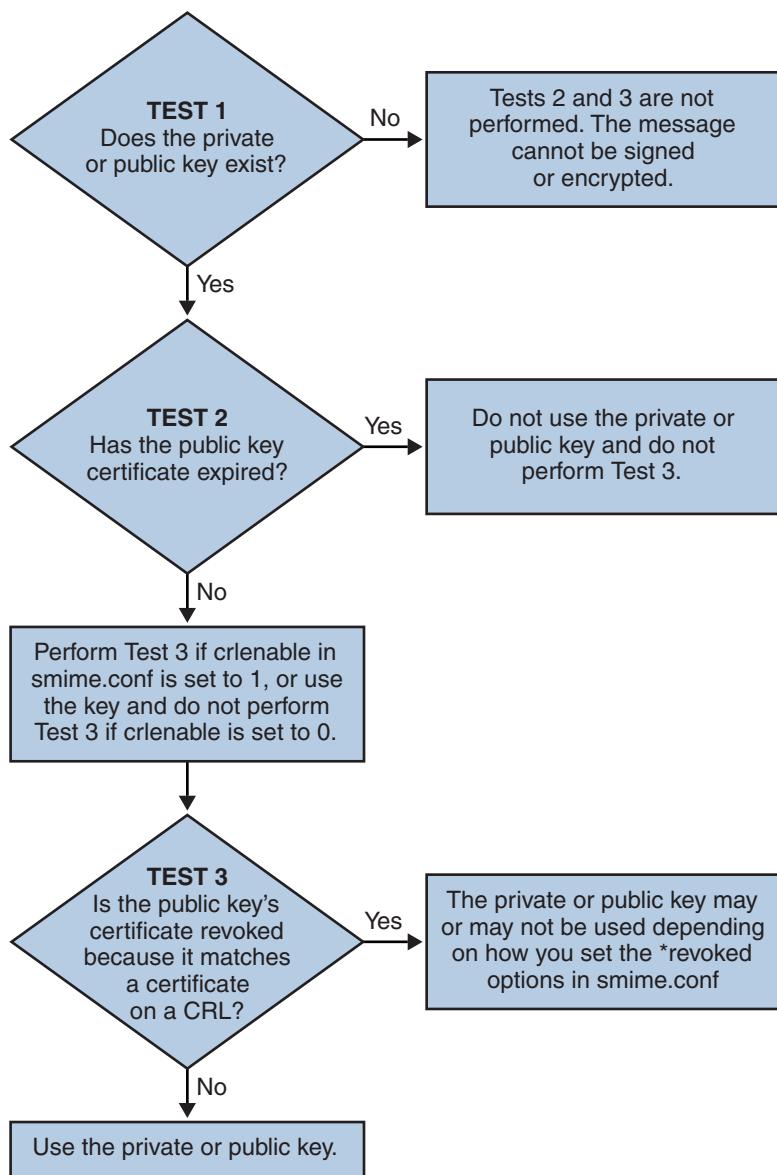


FIGURE 24–2 Verifying Private and Public Keys.

24.9.1

Finding a User’s Private or Public Key

When a Communications Express Mail user has multiple private-public key pairs and multiple email addresses (primary, alternate, or alias addresses), it is possible that their keys are

associated among their addresses. In this case, it is important that the S/MIME applet finds all the keys for verification purposes. Use the `usercertfilter` parameter in the `smime.conf` file to define a filter that creates a list of mail addresses for a key's owner at the time the public key's certificate is checked against a CRL. See `usercerfilter` in “[24.5 Parameters of the smime.conf File](#)” on page [764](#) for more information.

24.9.2 When is a Certificate Checked Against a CRL?

A certificate revocation list, or CRL, is a list of revoked certificates maintained by the CA who issues the key pairs and certificates. When CRL checking is enabled, it causes the system to check the CRL whenever a certificate request has been made to see whether or not that certificate has been revoked.

When `crlenable` is set to `1` in the `smime.conf` file, a CRL test is performed after an unexpired key is found. The public key's certificate is checked against a CRL. There can only be one CRL for each CA, however the same CRL can be located in different places.

Checking a certificate against a CRL is done by the Messaging Server after the S/MIME applet sends it a request to do so. A public key certificate is used to validate a public key. Because a private key is kept secret, only used by the person who owns it, a private key cannot be checked directly against a CRL. To determine if a private key is good, the public key certificate of the key pair is used. When the public key's certificate passes the CRL test, the associated private key passes the test too.

Revocation of a certificate can happen for a variety of reasons, such as its owner has left your organization or lost the smart card.

There are three situations for checking a certificate against a CRL:

- When an outgoing message is signed

The S/MIME applet always does this check unless you set `sendsigncert` to `0` or `crlenable` to `0`.

- When an incoming signed message is read

The S/MIME applet always does this check unless you set `readsigncert` to `0` or `crlenable` to `0`.

- When an outgoing message is encrypted

The S/MIME applet always does this check unless you set `sendencryptcert` to `0` or `crlenable` to `0`.

24.9.3 Accessing a CRL

A certificate contains zero or more URLs, known as distribution points, that are used by Messaging Server to locate a CRL. If the certificate does not have a CRL URL, it cannot be checked against a CRL and the private or public key is used to sign or encrypt a message without knowing its true status.

If Messaging Server fails to locate or gain access to a CRL after trying all the URLs available to it, the status of the certificate is treated as unknown. Whether a private or public key with an unknown status is used is determined by the setting of `revocationunknown`.

While only one CRL for each CA is supported, there can be multiple copies of the same CRL in different locations, reflected in different URLs among a user's public key certificates. Messaging Server tries all the URL locations for a certificate until it gains access to the CRL.

You can manage multiple copies of a CRL for optimum access by periodically downloading the current CRL from the CA to a place where you want it. While you cannot change the URLs embedded in the certificates, you can redirect Messaging Server to use new CRL locations by mapping the URLs in a certificate to a new URL containing the CRL information. Create a list of one or more mapping definitions in the LDAP directory (see `crlmappingurl` in [Table 24-3](#)) with this syntax:

```
msgCRLMappingRecord=url_in_certificate==new_url[|url_login_DN|url_login_password]
```

`url_in_certificate` is the URL in the certificate containing the old information to locate the CRL. `new_url` is the new URL containing the new CRL information. `url_login_DN` and `url_login_password` are the DN and password of the entry allowed access to `new_url`. Both are optional, and if specified, will be used for the new URL access only.

If the DN and password fails, LDAP access is denied and no retry with other credentials is attempted. These login credentials are only valid for LDAP URLs. If you use `crlurllogindn` and `crlurlloginpw` in `smmime.conf`, then you don't need to specify the login DN and password in the mapping record. See [“24.4.3 Accessing LDAP for Public Keys, CA certificates and CRLs Using Credentials” on page 762](#)

Only one layer of mapping is allowed. Different URLs in the certificates can be mapped to the same new URL, but you cannot assign a certificate URL to multiple new URLs. For example, the following mapping list is not valid:

```
msgCRLMappingRecord=URL12==URL45
msgCRLMappingRecord=URL12==URL66
msgCRLMappingRecord=URL12==URL88
msgCRLMappingRecord=URL20==URL90
msgCRLMappingRecord=URL20==URL93
```

The next example is a correct mapping list:

```
msgCRLMappingRecord=URL12==URL45
msgCRLMappingRecord=URL14==URL66
msgCRLMappingRecord=URL88==URL66
msgCRLMappingRecord=URL201==URL90
msgCRLMappingRecord=URL202==URL93
```

Once you have created the mapping definitions in your LDAP directory, use `crlmappingurl` in the `smime.conf` file to specify the directory information to locate them. See “[24.5 Parameters of the smime.conf File](#)” on page [764](#).

24.9.4 Proxy Server and CRL Checking

If your system uses a proxy server between client applications and the Messaging Server, CRL checking can be blocked despite the fact that you correctly configured the S/MIME applet to perform CRL checking. When this problem occurs, users of Communications Express Mail receive error messages alerting them to revoked or unknown status for valid key certificates.

The following conditions cause the problem:

- CRL checking is requested with these configuration values:
 - `crlenable` parameter in the `smime.conf` file is set to 1
 - `local.webmail.cert.enable` option of Messaging Server is set to 1
- The communications link between the S/MIME applet and the proxy server is not secured with SSL, but the S/MIME applet is expecting a secured link because the `checkoverssl` parameter in the `smime.conf` file is set to 1

To solve this problem, you can:

1. Set up the communications link between the client machines and proxy server as a secured link with SSL and leave all the configuration values as they are. Or,
2. Leave the communications link unsecured and set `checkoverssl` to 0.

For more information see “[24.7 Securing Internet Links With SSL](#)” on page [771](#).

24.9.5 Using a Stale CRL

Checking a certificate against a CRL is done by the Messaging Server after the S/MIME applet sends it a request to do so. Rather than download a CRL to memory each time a certificate is checked, Messaging Server downloads a copy of the CRL to disk and uses that copy for certificate checking. Every CRL has a next-update field which specifies the date after which a newer CRL version should be used. The next-update date can be viewed as an expiration date or time limit for using the CRL. A CRL that is past its next-update date is considered old or stale and triggers Messaging Server to download the latest version of the CRL the next time a certificate is checked.

Every time the S/MIME applet requests that a certificate be checked against a CRL, the Messaging Server does the following:

1. Compares the current date to the next-update date of the CRL.
2. If the CRL is stale, the Messaging Server downloads the latest version of the CRL to replace the stale CRL on disk and checking proceeds. However, if a newer CRL cannot be found or cannot be downloaded, the value of `crlusepastnextupdate` in the `smime.conf` file is used to determine what to do.
3. If `crlusepastnextupdate` is set to `0`, the stale CRL is not used and the certificate in question has an ambiguous status. The S/MIME applet uses the value of `revocationunknown` in `smime.conf` to determine what to do next:
 - a. If `revocationunknown` is set to `ok`, the certificate is treated as valid and the private or public key is used to sign or encrypt a message.
 - b. If `revocationunknown` is set to `revoked`, the certificate is treated as invalid, the private or public key is not used to sign or encrypt a message, and a pop-up error message alerts the mail user that the key cannot be used.

If `crlusepastnextupdate` is set to `1`, the S/MIME applet continues to use the stale CRL which causes no interruption of processing within Communications Express Mail, however a message is written to the Messaging Server log file to alert you to the situation.

This sequence of events continues to occur as certificates are checked against the CRL. As long as the Messaging Server can download a newer version of the CRL in a timely manner, and depending on the settings in the `smime.conf` file, mail processing proceeds without interruption. Check the Messaging Server log periodically for repeated messages that indicate a stale CRL is in use. If a newer CRL cannot be downloaded, you need to investigate why it is inaccessible.

24.9.6 Determining Which Message Time to Use

The `timestampdelta` parameter is used primarily for these purposes:

1. To handle the situation of a message that takes a long time to arrive at its destination. For this case, the sender's key might be treated as an invalid key despite the fact that the key was valid when the message was sent.
2. To limit the trust in a message's sent time because sent times can be faked.

There are two times associated with every message:

- The time when the message was sent, as found in the Date line of the message header detail
- The time when the message arrives at its destination, as found in the last Received line of the message header detail

Note – View the message header detail by clicking the triangle icon at the right hand side of a message's From field.

A certificate that was valid when a message was sent can be revoked or expired by the time the message reaches its destination. When this happens, which time should be used when checking the validity of the certificate, the sent time or the received time? Using the sent time would verify that the certificate was valid when the message was sent. But always using the sent time does not take into account the fact that it might take a long time for a message to arrive at its destination, in which case it would be better to use the received time.

You can influence which time to use for CRL checking by using the `timestampdelta` parameter in the `smime.conf` file. Set this parameter to a positive integer, representing seconds. If the received time minus the value of `timestampdelta` is a time before the sent time, the sent time is used. Otherwise, the received time is used. The smaller the value of `timestampdelta`, the more often the received time is used. When `timestampdelta` is not set, the received time is always used. See `timestampdelta` in [Table 24–3](#).

24.9.7

Trouble Accessing a CRL

For a variety of reasons, such as network or server problems, a CRL might be unavailable when Messaging Server attempts to check a certificate against it. Rather than let the Messaging Server spend its time constantly trying to gain access to the CRL, you can use the `crlaccessfail` parameter in the `smime.conf` file to manage how often it attempts to access the CRL, freeing up the Messaging Server for other tasks.

Define the following with `crlaccessfail`:

- How many failed attempts are counted (an error message is written to the Messaging Server log after each failed attempt)
- Over what period of time the failed attempts are counted
- How long to wait before attempting a new cycle of accessing the CRL

See `crlaccessfail` in [Table 24–3](#) for the parameter's syntax and an example.

24.9.8

When a Certificate is Revoked

When a public key's certificate does not match any entry on the CRL, the private or public key is used to sign or encrypt an outgoing message. When a certificate matches an entry on the CRL or the certificate's status is unknown, a private or public key is considered revoked. By default Communications Express Mail does not use a key with a revoked certificate to sign or encrypt an outgoing message. If the private key of a signed message is revoked by the time the recipient reads the message, the recipient receives a warning message indicating that the signature should not be trusted.

If desired, you can change the various default policies for all revoked certificates with the following parameters in the `smime.conf` file:

- Set `sendsigncertrevoked` to `allow` to sign an outgoing message with a private key that is considered revoked because its public key's certificate is revoked
- Set `sendencryptcertrevoked` to `allow` to encrypt an outgoing message with a public key that has a revoked certificate
- Set `revocationunknown` to `ok` to treat a certificate as valid whose status is unknown; the private or public key is used to sign or encrypt an outgoing message

24.10 Granting Permission to Use S/MIME Features

Permission to use the various mail services available through Communications Express Mail can be given or denied with LDAP filters. A filter is defined with the `mailAllowedServiceAccess` or `mailDomainAllowedServiceAccess` LDAP attributes. Generally speaking, a filter works in one of three ways:

- Permission is given to all users for all services when no filter is used
- Permission is explicitly given to a list of users for specified service names (a plus sign (+) precedes the service name list)
- Permission is explicitly denied to a list of users for specified service names (a minus sign (-) precedes the service name list)

The required mail service names for S/MIME are `http`, `smime`, and `smtp`. If you need to restrict the use of S/MIME among Communications Express Mail users, use the appropriate LDAP attribute syntax and service names to create a filter. The attributes are created or modified with LDAP commands.

24.10.1 S/MIME Permission Examples

1. The following examples block access to the S/MIME features for one Communications Express Mail user:

```
mailAllowedServiceAccess: -smime:*$+imap,pop,http,smtp:*
```

or

```
mailAllowedServiceAccess: +imap,pop,http,smtp:*
```

2. The following examples block access to the S/MIME features for all Communications Express Mail users in a domain:

```
mailDomainAllowedServiceAccess: -smime:*$+imap:$+pop:$+smtp:$+http:*
```

or

```
mailDomainAllowedServiceAccess: +imap:$+pop:$+$smtp:$+$http:$
```

See “[23.7.2 Filter Syntax](#)” on page 739 for more information.

24.11 Managing Certificates

Most of the following examples use the `ldapsearch` and `ldapmodify` commands to search an LDAP directory for user keys and certificates. These commands are provided with Directory Server. See the *Sun ONE Directory Server Resource Kit Tools Reference*, Release 5.2, for more information about the commands.

24.11.1 CA Certificates in an LDAP Directory

This example adds a certificate for a certificate authority to an LDAP directory. The directory structure for these certificates already exists. The certificate and the LDAP entries where it belongs are entered into an `.ldif` file named `add-root-CA-cert.ldif`. All text is entered into the file in ASCII text except for the certificate information, which must be entered as Base64 encoded text:

```
dn: cn=SMIME Admin,ou=people,o=demo.siroe.com,o=demo
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: certificationAuthority
cn: RootCACerts
sn: CA
authorityRevocationList: novalue
certificateRevocationList: novalue
cacertificate;binary:: MFU01JTUUeJaqBgNVBAsTCU1zZ1NlcnjcMBoGA1UEAxMTydg
QGEwJVUzEOMAwGA1UEMFUJTUUxEjAQBgNVBAsTCU1zZ1NlcnjcEMBoGA1UEAxMTQ2VydG
aFw0vNjAxMwODAwMDBaM267hgbx9FEcCzAJByrjgNVBAk9STklBMQwCgYDVQQVHR8EgaQwg
YTAlVMMRQYDVQQIEwpDQUxJRK9STklBMQwCgYDVQQKEww3ltgYz11lzAdBgNVBpYSE9Vc
5yZwQaddWlm899XBsYW5ldC5jb20wgZ8wDQYJoGBAK1mUTy8vn0Fg4mlHjkghytQUR1k8l
5mvWRf77ntm5mGXRD3XMU40ciUq6zUfIg3ngvxLyERTIqjUS8HQU4R5pvj+rrVgsAGjggE
+FNAJmtOV2A3wMyghqkVPNDP3Aqq2fkcn4va3C5nRNAYXNNVE84JJ0H3jyPDxhMBl0U6vQn
weMBAAjggEXMIIBezARBglghkgBhCAQEBApqlSai4mfuvjh02SQkoPMNDAgTwMB8GA1UdI
QYMBaAEd38IK05AHreiU90Yc6vNM0wZMIGsBgnVHR8EgaQwgaEwb6BtoGuGaWxkYXA6Lyht
bmcucmVkLmlbGFuZXQuY29tL1VJD1DXJ0awZpY2F0ZSBNYW5hZ2VyLE9VPVBb3BsZsPPW
aWxxYT9jZXJ0aZpY2jdu2medXRllkghytQURYFNrkuoCygKoYoaHR0cDovL3Bla2kghytQU
Zy5yZwQuaXBsYW5lC5jb20vcGVranLmNybdAeBgNVHREEFzAVgRnwB3J0aWEuc2hhb0BzdW
4uY29tMA0GCxLm78freCxS3Pp078jyTaDci1AudBL8+RrRUQvxsmjfZeFED+Uuf10Ilt6kw
Tc6W5UekbirfEZGAVQIzlt6DQJfgpifGLvtQ60Kw==
```

The CA's certificate is added to the LDAP directory with an `ldapmodify` command:

```
# ldapmodify -a -h demo.siroe.com -D "cn=Directory Manager" -w mypasswd -v
-f add-root-CA-cert.ldif
```

The value of the `trustedurl` parameter in `smime.conf` specifies the location of the CA certificates in the LDAP directory. For Example 1, `trustedurl` is set to:

```
trustedurl==ldap://demo.siroe.com:389/cn=SMIME Admin, ou=people,
o=demo.siroe.com,o=demo?cacertificate;binary?sub?
(objectclass=certificationAuthority)
```

24.11.2 Public Keys and Certificates in an LDAP Directory

This example demonstrates adding a mail user's public key and certificate to the LDAP directory. It assumes the mail user already exists in the LDAP directory. The key and certificate, and the LDAP entries where it belongs, are entered into an `.ldif` file named `add-public-cert.ldif`. All text is entered into the file as ASCII text except for the key and certificate information, which must be entered as Base64 encoded text.

```
dn: uid=JohnDoe,ou=People, o=demo.siroe.com,o=demo
changetype: modify
replace: usercertificate
usercertificate;binary:: MFU01JTUUxEjAQBgNVBAsT1zZ1NlcnZlcjMBoGA1UEAxMTydg
QGEwJVUzEAwGA1hMFU01JTUUxEjAQBgNVBAsTCU1zZ1NlcnZlcjEcMBoGA1UEAxMTQ2VydG
aFw0wNjAxMTODAwM267hgbX9FExCzAjBgwyrgjnvBAk9STklBMQwwCgYDVQQVHR8EgaQwg
AlVzMRMwEQYDVQQIDQUxJRK9STklBMQwwCgYDVQQKEww3ltgoOYZ11lzAdBgNVBpYSE9Vc
5yZWaddiiWlm899XBsYW5ldb20wgZ8wDQYJoGBAk1mUTy8vv02n0Fg4mlHjkghytQUR1k8l
5mvgcWL77ntm5mGXRD3XMU40cizUfIg3ngvxllKLjERTIqjUS8HQU4R5pvj+rrVgsAGjggE
+FG9NAqtOV2A3wMyghqkVPNDP3Aqq2Yfkcn4va3RNAYxNNVE84JJ0H3jyPDXhMB1lQU6vQn
1NAgMBGjggEXMIIBEZARBglghkgBvhCAQEEBapqlSai4mfvjh02SQMNDAgTwMB8GA1UDI
QYMBaEd38IK05AHreiU90Yc6v+ENMOwZMIGsBgnVHR8EgaQwgaEwb6BuGaWxkYXA6Lyht74
tpbcmvKLMlwbgFUZXQuY29tL1VJRD1DZXJ0aWZpY2F0ZSBNYw5hZ2V9VPVBlb3BsZSxPPW
1haWxT9jZXJ0aWZpY2jdu2medXR1lHjkghytQURYFNrkuoCygKoYoaHDovL3Bla2kghytQU
luZy5WQuaXBsYW5ldC5jb20vcGVraW5nLmNyBdAeBgnVHREEFzAvgrNw0aWEuc2hhb0BzdW
4uY29A0GCxLm78UfreCxS3Pp078jyTaDv2ci1AudBL8+RrRUQvxsmJfZD+Uuf10Ilt6kwhm
Tc6W5UekbirfEZGAVQIzlt6DQJfgpifGLvtQ60Kw==
```

The `ldapmodify` command is used to add the public key and certificate to the LDAP directory:

```
# ldapmodify -a -h demo.siroe.com -D "cn=Directory Manager" -w mypasswd -v
-f add-public-cert.ldif
```

The value of the `certurl` parameter in `smime.conf` specifies the location of the public keys and their certificates in the LDAP directory. For Example 2, `certurl` is set to:

```
certurl==ldap://demo.siroe.com:389/ou=people, o=demo.siroe.com,  
o=demo?userCertificate;binary?sub?
```

24.11.3 Verifying That Keys and Certificates Exist in the LDAP Directory

The following examples demonstrate searching an LDAP directory for CA certificates and public keys and their certificates.

24.11.3.1 Searching for One CA Certificate

In the following example, the base DN defined by the `-b` option, `cn=SMIME admin, ou=people, o=demo.siroe.com, o=demo`, describes one CA certificate in the LDAP directory. If found in the directory, `ldapsearch` returns information about the certificate to the `ca-cert.ldif` file.

```
# ldapsearch -L -h demo.siroe.com -D "cn=Directory Manager" -w mypasswd -b  
"cn=SMIME admin, ou=people, o=demo.siroe.com, o=demo" "objectclass=*"  
> ca-cert.ldif
```

The example below shows the search results in the `ca-cert.ldif` file. The format of the file's contents is a result of using the `-L` option of `ldapsearch`.

```
# more ca-cert.ldif  
dn: cn=SMIME admin,ou=people,o=demo.siroe.com,o=demo  
objectClass: top  
objectClass: person  
objectClass: organizationalPerson  
objectClass: inetOrgPerson  
objectClass: certificationAuthority  
cn: RootCACerts  
cn: SMIME admin  
sn: CA  
authorityRevocationList: novalue  
certificateRevocationList: novalue  
cacertificate;binary:: MFU0JITUUxEjAQBgNVBAsTCU1zZNlcjnZlcjcMBoGA1UEAxMTydg  
QGEwJVEOMAwGA1UEChMFU0UUxEjAQBgNVBAsTCU1zZ1NlcjnZlcjEcMBoGA1UEAxMTQ2VydG  
aFw0jAxMTIwODAwMDBaM267X9FExCzAJBgwyrgjgNVBAk9STklBMQwwCgYDVQQVHR8EgaQwg  
YlVzMrmwEQYDVQQIEwpDQUx9STklBMQwwCgYDVQQKEww3ltgoOYz11lzAdBgNVBpYSE9Vc  
5yQuaddiiWlm899XBsYW5ljb20wgZ8wDQYJoGBAK1mUTy8vv02nOFg4mlHjkghytQUR1k8l  
5mcWRfL77ntm5mGXRDXMcIuq6zUfIg3ngvxllKLlyERTIqjUS8HQU4R5pvj+rrVgsAGjggE  
+FNAJmqtOV2A3wMyghqkDP3Aqq2BYfkcn4va3C5nRNAYxNNVE84JJ0H3jyPDxhMBlQU6vQn  
1NABAAGjggEXMIBEzglghkgBvhCAQEEBApqlSai4mfuvjh02SQkoPMNDAgTwMB8GA1UdI  
QYMAFEd38IK05AHre0Yc6v+ENMOwZMIGsBgNVHR8EgaQwgaEwb6BtoGuGaWxkYXA6Lyht74  
tpbucmVkLmlwbGFuZY29tL1VJRD1DZXJ0awZpY2F0ZSBNYW5hZ2VyLE9VPVBlb3BsZSxPPW
```

```
1haWYT9jZXJ0aWZpdu2medXRllHjkghytQURYFNrkuoCygKoYoaHR0cDovL3Bla2kghytQU
luZyZWQuaXBsYW5ldb20vcGVraW5nLmNybDAeBgNVHREEFzAVgRNwb3J0aWEuc2hhb0BzdW
4UYtMA0GCxLm78Ufre3Pp078jyTaDv2ci1AudBL8+RrRUQvxsmJfZeFED+Uuf10Ilt6kwhm
Tc6W5UekbirfEZGAVQIzlt6DQJfgpifGLvtQ60Kw==
```

Searching for a Several Public Keys

In the following example, the base DN defined by the `-b` option, `o=demo.siroe.com, o=demo` `objectclass=*`, is such that all public keys and certificates found at and below the base DN in the LDAP directory are returned to the file `usergroup.ldif`:

```
# ldapsearch -L -h demo.siroe.com -D "cn=Directory Manager" -w mypasswd
-b "o=demo.siroe.com, o=demo" "objectclass=*" > usergroup.ldif
```

Searching for One Public Key

In the following example, the base DN defined by the `-b` option, `uid=JohnDoe, ou=people, o=demo.siroe.com, o=demo` `objectclass=*`, describes one public key and its certificate in the LDAP directory:

```
# ldapsearch -L -h demo.siroe.com -D "cn=Directory Manager" -w mypasswd -b
"uid=JohnDoe, ou=people, o=demo.siroe.com, o=demo" "objectclass=*" > public-key.ldif
```

The example below shows the search results in the `public-key.ldif` file. The format of the file's contents is the result of using the `-L` option of `ldapsearch`.

```
# more public-key.ldif
dn: uid=sdemo1, ou=people, o=demo.siroe.com, o=demo
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: siroe-am-managed-person
objectClass: inetOrgPerson
objectClass: inetUser
objectClass: ipUser
objectClass: userPresenceProfile
objectClass: inetMailUser
objectClass: inetLocalMailRecipient
objectClass: icsCalendarUser
objectClass: sunUCPreferences
mail: JohnDoe@demo.siroe.com
mailHost: demo.siroe.com
.
.
uid: JohnDoe
.
.
```

```
mailUserStatus: active
inetUserStatus: active
.
.
usercertificate;binary:: MFU01JTUUxEjAQBgNBAsTCU1zZ1NlcnZjcMBoGA1UEAxMTydg
QGEwJE0wGA1UEChMFU01JTUUxEjAQBgNVBAsTCU1zZ1NlcnZlcjEcMBoGA1UEAxMTQ2VydG
aFw0MTIwODAwMDBaM267hbX9FEExCzAJBgwyrgjgNVBAk9STklBMQwwCgYDVQQVHR8EgaQwg
YTAlVEQYDVQQIEwpDQUxJRK9STklBMQwwCgYDVQQKEww3ltgoOYz11lzAdBgNVBpYSE9Vc
5yZWQdWlm899XBsYW5ldC5jb20wgZ8wDQYJoGBAK1mUTy8vv02n0Fg4mlHjkghytQUR1k8l
5mvgc7ntm5mGXRDXMU40ciUq6zUfIg3ngvxLKLKyERTIqjUS8HQU4R5pvj+rVgsAGjggE
+FG9NmV2A3wMyghqkVPNDP3Aqq2BYfkcn4va3C5nRNAYxNNVE84JJ0H3jyPDXhMB1QU6vQn
1NAgMAgEXMIIBezARBg1ghkgBhvhaCAQEEBApqlSai4mfuvjh02SQkoPMNDAgTwMB8GA1UdI
QYMBaEdK05AHreiU90Yc6v+ENMOwZMIGsBgvNVR8EgaQwgEwB6BtoGuGaWxkYXA6Lyht74
tpbucmVkbwGFuZXQuY29tL1VJRD1DZXJ0awZpY2F0ZSBNYW5hZ2VyLE9VPVBlb3BsZsxPPW
1haxYT9jZaWZpY2jdu2medXRllHjkghytQURYFNrkuoCygKoYoaHR0cDovL3Bla2kghytQU
luZyZWQuYW5ldC5jb20vcGvraW5nLmNybdAeBgNVHREEFzAVgRNwb3J0aWEuc2hhb0BzdW
4u9tMA0GC78UfreCxS3Pp078jyTaDv2ci1AudBL8+RrRUQvxsMJfZeFED+Uuf10Ilt6kwhm
Tc6W5UekbirfEZGAVQIzlt6DQJfgpifGLvtQ60Kw==
```

24.11.4 Network Security Services Certificates

Various certificates used for Network Security Services (NSS) are stored in their own database, which is not an LDAP database. Two utilities, `certutil` and `crlutil`, are provided with Messaging Server to store the certificates and associated CRLs in the database. You can also use these utilities to search the database.

See the *Sun Java System Directory Server Administration Guide* (<http://docs.sun.com/doc/817-7613>) for more information about `certutil`. Use the help text that comes with `crlutil` for more information about that utility (view the online help of both utilities by executing them without arguments).

24.12 Communications Express S/MIME End User Information

This section consists of information intended for the end user. It contains the following subsections:

- “24.12.1 Logging In for the First Time” on page 787
- “24.12.2 Signature and Encryption Settings” on page 788
- “24.12.3 Enabling the Java Console” on page 789

24.12.1 Logging In for the First Time

When mail users log in to Communications Express Mail for the first time, they encounter special prompts relating to the S/MIME applet.

24.12.1.1 Prompts for Windows

When logging in to Communications Express Mail for the first time on Windows 98, 2000 or XP, the following prompts display:

1. If the Java 2 Runtime Environment (JRE) is not installed on your computer (client machine), you receive a prompt looking something like this:

Do you want to install and run "Java Plug-in 1.4.2_03 signed on 11/20/03 and distributed by Sun Microsystems, Inc."? Publisher authenticity verified by: VeriSign Class 3 Code Signing 2001 CA

Click Yes and follow the subsequent prompts to install JRE.

Note – If you desire English language support and also want to read incoming S/MIME messages that contain non-Latin characters, such as Chinese, the `charsets.jar` file must be in the `/lib` directory on your computer.

To ensure that the `charsets.jar` file is installed in the `/lib` directory, use the custom installation to install the English version of JRE. During the installation process, select the “Support for Additional Languages” option.

See “[24.3.6 Multi-language Support](#)” on page 756 for more information.

Click Finish at the last installation prompt. Restart your computer and log in to Communications Express Mail again.

2. A prompt asking you:

Do you want to trust the signed applet distributed by "Sun Microsystems, Inc."? Publisher authenticity verified by: Thawte Consulting cc

Click one of the following responses:

- Yes, to accept the S/MIME applet for this Communications Express Mail session. The prompt displays each time you log in.
- No, to reject the S/MIME applet. You cannot use the S/MIME features.
- Always, to accept the S/MIME applet for this and all subsequent Communications Express Mail sessions. You will not see the prompt again.

3. A prompt asking you:

Do you want to trust the signed applet distributed by "sun microsystems, inc."? Publisher authenticity verified by: VeriSign, Inc.

Click one of the following responses:

- Yes, to accept the S/MIME applet for this Communications Express Mail session. The prompt displays each time you log in.
- No, to reject the S/MIME applet. You cannot use the S/MIME features.
- Always, to accept the S/MIME applet for this and all subsequent Communications Express Mail sessions. You will not see the prompt again.

24.12.2 Signature and Encryption Settings

There are initial signature and encryption settings that you can set to control whether all users' outgoing messages are:

- automatically signed, or
- automatically encrypted, or
- automatically signed and encrypted

The initial settings also control whether the signature and encryption checkboxes located at the bottom of a Communications Express Mail window and in the Options - Settings window are displayed as checked (feature turned on) or unchecked (feature turned off). Use the `alwaysencrypt` and `alwaysign` parameters in the `smime.conf` file to specify the initial settings.

Let your mail users know that they can change the initial settings for their mail messages. After they log in to Communications Express Mail, a user can temporarily override a setting for one message, or for all their messages on an on-going basis.

[Table 24–5](#) summarizes the use of the checkboxes.

TABLE 24–5 Signature and Encryption Checkboxes of Communications Express Mail

Text for Checkbox	Location	What Communications Express Mail User Does
Sign Message	At the bottom of the Communications Express Mail window used for composing, forwarding, or replying to a message.	<ul style="list-style-type: none">■ Check the box to sign the current message.■ Uncheck the box not to sign the current message.
Encrypt Message	At the bottom of the Communications Express Mail window used for composing, forwarding, or replying to a message.	<ul style="list-style-type: none">■ Check the box to encrypt the current message.■ Uncheck the box not to encrypt the current message.

TABLE 24–5 Signature and Encryption Checkboxes of Communications Express Mail *(Continued)*

Text for Checkbox	Location	What Communications Express Mail User Does
Sign all outgoing Messages	In the Communications Express Mail Options - Settings window, under the Secure Messaging option.	<ul style="list-style-type: none"> ■ Check the box to sign all your messages automatically. ■ Uncheck the box not to sign all your messages automatically. Note: You can override the setting of “Sign all outgoing messages” on a message-by-message basis with the “Sign Message” checkbox.
Encrypt all outgoing Messages	In the Communications Express Mail Options - Settings window, under the Secure Messaging option.	<ul style="list-style-type: none"> ■ Check the box to encrypt all your messages automatically ■ Uncheck the box not to encrypt all your messages automatically. Note: You can override the setting of “Encrypt all outgoing messages” on a message-by-message basis with the “Encrypt Message” checkbox.

24.12.3 Enabling the Java Console

A variety of operating messages can be written to the Java Console by the S/MIME applet as a Communications Express Mail user processes signed and encrypted messages. The Java Console messages can be helpful when troubleshooting a problem reported by a mail user. However, operating messages are only generated when the Java Console is enabled for the user by adding a `nswmExtendedUserPrefs` attribute to the `inetMailUser` object class of their LDAP entry. For example:

```
nswmExtendedUserPrefs: meSMIMEDebug=on
```

Do not enable the Java Console for all mail users all the time because this significantly decreases the performance of Communications Express Mail.

Managing Logging

This chapter provides overview information on the logging facilities for the Messaging Server MTA, the Message Store, and services. This chapter also provides procedures for how to manage these logging facilities.

This chapter contains the following sections:

- “[25.1 Overview of Logging](#)” on page 791
- “[25.2 Tools for Managing Logging](#)” on page 795
- “[25.3 Managing MTA Message and Connection Logs](#)” on page 796
- “[25.4 Managing Message Store, Admin, and Default Service Logs](#)” on page 820

25.1 Overview of Logging

Logging is the means by which a system provides you with time-stamped and labeled information about the system’s services. Logging provides both a current snapshot of the system as well as a historical view.

By understanding and using Messaging Server log files, you can:

- Gather message statistics, such as message size, rate of message delivery, and how many messages are passing through the MTA
- Perform trend determination
- Correlate capacity planning
- Troubleshoot problems

For example, if your site needs to add more disk storage due to an increase in the number of users, you can use the Messaging Server log files to see what percentage your system demand has increased by and plan for the amount of new disk storage you need.

You can also use Messaging Server logs to understand what your messaging pattern looks like across one day. Understanding when your daily peak loads occur helps you conduct capacity planning.

Logging is also helpful for troubleshooting user problems. For example, if a user isn't receiving expected mail messages, you can use the Messaging Server logging facilities to trace the user's mail messages. In so doing, you might find out that the messages didn't arrive because they were automatically filtered and sent to a SPAM folder.

25.1.1 Types of Logging Data

In general, logging provides you with two types of information:

- Operational data
- Error conditions, also known as event logging

For the most part, Messaging Server logging provides operational data. This operational data contains information such as: the date and time a message entered the system; the sender and recipient of the message; when the message was written to disk; and at a later point in time, when the message was removed from disk and inserted into user's mailbox.

However, Messaging Server logging does also provide some event logging data. To obtain event logging data, you need to pull together multiple items from different log files. You could then use a unique constant, such as message ID, to search and correlate the lifecycle of a message as it passed from point to point through the system.

25.1.2 Types of Messaging Server Log Files

Messaging Server logging consists of three types of log files:

1. **MTA logs.** These logs provide operational data previously described for the Message Transfer Agent.
2. **Error logs.** These are the MTA debug logs, and the MTA subcomponent logs (that is, job controller, dispatcher and so on).
3. **Message Store and Service logs.** These logs provide messages from the http server, mshttpd, imap, and pop services, as well as the Admin service. The format of these logs differs from that of the first two types of logs.

The following table lists the different types of log files. By default, log files are located in the *msg-svr-base*/data/log directory. You can customize and view each type of log file individually.

TABLE 25-1 Messaging Server Log Files

Type of Log File	Log File description	Default Name
Message Transfer Agent	Show information about message traffic through the MTA including date and time information, enqueue and dequeue information, and so on.	mail.log, mail.log_current, mail.log_yesterday
Connections	Contains remote machines (MTAs) that connect to this system to send email.	connection.log
Counters	Contains message trends in terms of messages sent and received on a per channel basis.	counters
Job Controller	Contains data on the master, job controller, sender, and dequeue channel programs.	job_controller.log
Dispatcher	Contains errors pertaining to the dispatcher. Turning on dispatcher debugging will increase the information.	dispatcher.log
Channel	Records errors pertaining to the channel. Keywords master_debug and slave_debug turns on channel debugging, which increases the verbosity of the channel log files. Level and type of information is controlled with the various *_DEBUG MTA options in option.dat.	<i>channelname_master.log*</i> (example: <i>tcp_local_master.log*</i>) <i>channelname_slave.log*</i> (example, <i>tcp_local_slave.log*</i>)
IMAP	Contains logged events related to IMAP4 activity of this server	imap, imap.sequenceNum.timeStamp
POP	Contains logged events related to POP3 activity of this server	pop, pop.sequenceNum.timeStamp
HTTP	Contains logged events related to HTTP activity of this server	http, http.sequenceNum.timeStamp
Default	Contains logged events related to other activity of this server, such as command-line utilities and other processes	default, default.sequenceNum.timeStamp
msgtrace	Contains trace information for the Message Store. File can grow very large very quickly. Monitor accordingly.	msgtrace
watcher	monitor process failures and unresponsive services (see Table 4-4) and will log error messages indicating specific failures.	watcher

where:

sequenceNum - specifies an integer that specifies the order of creation of this log file compared to others in the log-file directory. Log files with higher sequence numbers are more recent than those with lower numbers. Sequence numbers do not roll over; they increase monotonically for the life of the server (beginning at server installation).

timeStamp - Specifies a large integer that specifies the date and time of file creation. (Its value is expressed in standard UNIX time: the number of seconds since midnight January 1, 1970.)

For example, a log file named `imap.63.915107696` would be the 63rd log file created in the directory of IMAP log files, created at 12:34:56 PM on December 31, 1998.

The combination of open-ended sequence numbering with a timestamp gives you more flexibility in rotating, expiring, and selecting files for analyzing. For more specific suggestions, see “[25.4.3 Defining and Setting Service Logging Options](#)” on page 824

25.1.3 Tracking a Message Across the Various Log Files

The following describes how a message flows through the system, and at what point information gets written to the various log files. This description is meant to aid you in your understanding of how to use Message Server’s log files to troubleshoot and resolve problems. See [Figure 8–2](#) to follow along.

1. A remote host makes a connection to the TCP socket on your messaging host, requesting SMTP service.
2. The MTA dispatcher responds to the request, and hands off the connection to your messaging host’s SMTP service.

As the MTA is modular in design, it consists of a set of processes, including the job controller and the SMTP service dispatcher. The dispatcher takes the incoming TCP connection and sends it to the SMTP service. The SMTP service writes the message to disk to a channel area. The SMTP service understands the message’s envelope parameters, such as sender and recipient. Configuration entries in the system tell what destination channel it belongs to.
3. The dispatcher writes to the `dispatcher.log` file that it forked a thread and made the thread available to incoming connection from a certain IP address.
4. The SMTP server writes to its `tcp_smtp_server.log` file, recording the dialog of what happens when the remote host connected to it and sent a message. This log file gets created when dispatcher hands off to SMTP server on the host’s IP.
5. The SMTP server writes the message to a queue area on disk for a channel program such as `tcp_intranet`, and informs the job controller.
6. The job controller contacts the channel program.
7. The channel program delivers the message.

Each channel has its own log file. However, these logs usually show the starting and stopping of the channel. To get more information, you need to enable debug level for the channel. However, as this can slow down your system and actually make problems more obscure if left on, you should only enable debug level when an actual problem is occurring.

Note – For efficiency, if a channel is already running for an existing process, and a new message comes in, the system does not spawn a new channel process. The currently running process picks up the new message.

8. The message is delivered to its next hop, which could be another host, another TCP connection, and so forth. This information is written in the `connection.log` file.

At the same time that the SMTP server writes the message to a queue area on disk, the channel responsible for the message writes a record in the `mail.log_current`, or `mail.log` file. The record shows such information as the date and time the message was enqueued, the sender, the recipient, so forth. See “[25.3.4 MTA Message Logging Examples](#)” on page 805 for more information. The most useful file for tracing the message is the `mail.log_current` file.

25.2 Tools for Managing Logging

You can customize the policies for creating and managing Messaging Server log files by using the `configutil` command.

For Message Store, the settings you specify affect which and how many events are logged. You can use those settings and other characteristics to refine searches for logged events when you are analyzing log files.

The MTA uses a separate logging facility you configure MTA logging by specifying information in configuration files.

For log analyses and report generation beyond the capabilities of Messaging Server, you need to use other tools. You can manipulate log files on your own with text editors or standard system tools.

With a scriptable text editor supporting regular-expression parsing, you can potentially search for and extract log entries based on any of the criteria discussed in this chapter, and possibly sort the results or even generate sums or other statistics.

In UNIX environments you might also be able to modify and use existing report-generation tools that were developed to manipulate UNIX `syslog` files. If you wish to use a public-domain `syslog` manipulation tool, remember that you might need to modify it to account for the different date/time format and for the two extra components (*facility* and *logLevel*) that appear in Messaging Server log entries but not in `syslog` entries.

25.3 Managing MTA Message and Connection Logs

The MTA provides facilities for logging each message as it is enqueued and dequeued. It also provides dispatcher error and debugging output.

This section consists of the following subsections:

- [“25.3.1 Understanding the MTA Log Entry Format” on page 796](#)
- [“25.3.2 Enabling MTA Logging” on page 800](#)
- [“25.3.3 Specifying Additional MTA Logging Options” on page 801](#)
- [“25.3.4 MTA Message Logging Examples” on page 805](#)
- [“25.3.5 Enabling Dispatcher Debugging” on page 818](#)

You can control logging on a per-channel basis or you can specify that message activity on all channels be logged. In the initial configuration, logging is disabled on all channels.

See [“25.3.2 Enabling MTA Logging” on page 800](#) for more information.

Enabling logging causes the MTA to write an entry to the *msg-svr-base/data/log/mail** file each time a message passes through an MTA channel. Such log entries can be useful for gathering statistics on how many messages are passing through the MTA (or through particular channels). You can also use these log entries to investigate other issues, such as whether and when a message was sent or delivered.

The message return job, which runs every night around midnight, appends any existing *mail.log_yesterday* to the cumulative log file, *mail.log*, renames the current *mail.log_current* file to *mail.log_yesterday*, and then begins a new *mail.log_current* file. The message return job also performs the analogous operations for any *connection.log** files.

While the MTA performs automatic rollovers to maintain the current file, you must manage the cumulative *mail.log* file by determining policies for tasks such as backing up the file, truncating the file, deleting the file, and so on.

When considering how to manage the log files, note that the MTA periodic return job will execute a site-supplied *msg-svr-base/bin/daily_cleanup* procedure, if one exists. Thus some sites might choose to supply their own cleanup procedure that, for instance, renames the old *mail.log* file once a week (or once a month), and so on.

Note – With logging is enabled, the *mail.log* file steadily grows and, if left unchecked, consumes all available disk space. Monitor the size of this file and periodically delete unnecessary contents. You can also delete the entire file as another version will be created as needed.

25.3.1

Understanding the MTA Log Entry Format

The MTA log file is written as ASCII text. By default, each log file entry contains eight or nine fields as shown in the example below.

```
16-Feb-2007 14:54:13.72 tcp_local ims-ms EE 1 adam@sesta.com
rfc822;marlowe@siroe.com marlowe@ims-ms-daemon
```

The log entry shows:

1. The date and time the entry was made (in the example, 16-Feb-2007 14:54:13.72).
2. The channel name for the source channel (in the example, `tcp_local`).
3. The channel name for the destination channel (in the example, `ims-ms`). (For SMTP channels, when `LOG_CONNECTION` is enabled, a plus (+) indicates inbound to the SMTP server; a minus (-) indicates outbound via the SMTP client.)
4. The type of entry (in the example, `EE`). Entries can consist of a single action code (see [Table 25-2](#)) or an action code and one or more modifier codes (see [Table 25-3](#)). The format for entries is as follows:

<action_code><zero or more optional modifiers>

For example a logging entry code of EEC means that the email was Enqueued (action-code E) using ESMTP (modifier E) and SMTP Chunking (modifier C). Please refer to the tables below for details on the currently used action and modifier codes.

5. The size of the message (in the example, 1). This is expressed in kilobytes by default, although this default can be changed by using the `BLOCK_SIZE` keyword in the MTA option file. The SMS channel can be configured to log a page count rather than file size in this field. See "[LOG_PAGE_COUNT](#)" on page [977](#).
6. The envelope `From:` address (in the example, `adam@sesta.com`). Note that for messages with an empty envelope `From:` address, such as notification messages, this field is blank.
7. The original form of the envelope `To:` address (in the example, `marlowe@siroe.com`).
8. The active (current) form of the envelope `To:` address (in the example, `marlowe@ims-ms-daemon`).
9. The delivery status (SMTP channels only).

The following three tables describe the logging entry codes.

TABLE 25-2 Logging Entry Action Codes

Entry	Description
B	Bad command sent to the SMTP server. The recipient address field will contain the command that was rejected while the diagnostic field will contain the response the SMTP server gave. MTA channel option, <code>MAX_B_ENTRIES</code> , controls how many bad commands will be logged in a given session. Default is 10.
D	Successful dequeue
E	Enqueue
J	Rejection of attempted enqueue (rejection by slave channel program)

TABLE 25–2 Logging Entry Action Codes (*Continued*)

Entry	Description
K	Recipient message rejected. If the sender requests NOTIFY=NEVER DSN flag set or if the message times out or if the message is manually returned (for example: <code>imsimta qm "delete"</code> command always generates a "K" record for each recipient, while a <code>qm "return"</code> command will generate a "K" record rather than an "R" record). This indicates that there was no notification sent to the sender per the sender's own request. This can be compared with "R" records, which are the same sort of rejection/time-out, but where a new notification message (back to the original sender) is also generated regarding this failed message.
Q	Temporary failure to dequeue
R	Recipient address rejected on attempted dequeue (rejection by master channel program), or generation of a failure/bounce message
V	Warning message that will appear whenever a transaction is abnormally aborted. There will be one "V" record per enqueued recipient address.
W	Warning message sent to notify original sender that the message has not been delivered yet, but it is still in the queue being retried.
Z	Some successful recipients, but this recipient was temporarily unsuccessful; the original message file of all recipients was dequeued, and in its place a new message file for this and other unsuccessful recipients will be immediately enqueued

The following table describes the logging entry modifier codes.

TABLE 25–3 Logging Entry Modifier Codes

Entry	Description
A	SASL authentication used.
C	Chunking was used. Note that ESMTP has to be used for chunking to work, so you'll typically see field values like EEC or DEC.
E	An EHLO command was issued/accepted and therefore ESMTP was used.
L	LMTP was used.
S	TLS/SSL used. S transaction log entries now increment the various submitted message counters associated with the channel.

If LOG_CONNECTION is enabled (see “Option File Format and Available Options” in *Sun Java System Messaging Server 6.3 Administration Reference*), then an additional set of action codes will be used. These are described below.

TABLE 25-4 SMTP Channel's LOG_CONNECTION Action Codes + or - Entries

Entry	Description
C	Connection closed. A diagnostic field will follow. Written to connection.log_current (or mail.log_current if a single log file is being used). Used to record the reason why the connection was closed. In particular, if the connection was closed due to some session disconnect limit being reached, that fact will show up in the diagnostics field.
O	Connection opened
U	Logs SMTP authentication successes and failures. Format is the same as other O and C entries. In particular, the same application and transport information fields appear in same order. The username will be logged in the username field if it is known. Bit 7 (value 128) of the LOG_CONNECTION MTA option controls this.
X	Connection rejected
Y	Connection attempt failed before being established
I	ETRN command received

With LOG_CONNECTION, LOG_FILENAME, LOG_MESSAGE_ID, LOG_NOTARY, LOG_PROCESS, and LOG_USERNAME all enabled in the MTA Option file, the format becomes as shown in the example below. (The sample log entry line has been wrapped for typographic reasons; the actual log entry would appear on one physical line.)

```
16-Feb-2007 15:04:01.14 2bbe.5.3 tcp_local ims-ms
EE 1 service@siroe.com rfc822;adam@sesta.com
adams@ims-ms-daemon 20 /opt/SUNWmsgsr/data/queue/ims-ms/000/ZZf0r2i0HIaY1.01
<0JDJ00803FAON200@mailstore.siroe.com> mailsrv
siroe.com (siroe.com [192.160.253.66])
```

Where the additional fields, beyond those already discussed above, are:

1. The process ID (expressed in hexadecimal), followed by a period (dot) character and a count. If this had been a multithreaded channel entry (that is, a `tcp_*` channel entry), there would also be a thread ID present between the process ID and the count. In the example, the process ID is 2bbe.5.3.
2. The NOTARY (delivery receipt request) flags for the message, expressed as an integer (in the example, 20).
3. The file name in the MTA queue area (in the example, `/opt/SUNWmsgsr/data/queue/ims-ms/000/ZZf0r2i0HIaY1.01`).
4. The message ID (in the example, `<0JDJ00803FAON200@mailstore.siroe.com>`).
5. The name of the executing process (in the example, `mailsrv`). On UNIX, for dispatcher processes such as the SMTP server, this will usually be `mailsrv` (unless SASL was used, in which case it will be the authenticated user name, for example, `*service@siroe.com`).

6. The connection information (in the example, `siroe.com` (`siroe.com [192.160.253.66]`)). The connection information consists of the sending system or channel name, such as the name presented by the sending system on the HELO/EHLO line (for incoming SMTP messages), or the enqueueing channel's official host name (for other sorts of channels). In the case of TCP/IP channels, the sending system's real name, that is, the symbolic name as reported by a DNS reverse lookup and/or the IP address, can also be reported within parentheses as controlled by the `ident*` channel keywords; see “[12.4.3.4 IDENT Lookups](#)” on page 372 for an instance of the default `identnone` keyword, that selects display of both the name found from the DNS and IP address.

25.3.2 Enabling MTA Logging

To gather statistics for just a few particular MTA channels, enable the logging channel keyword on just those MTA channels of interest. Many sites prefer to enable logging on all MTA channels. In particular, if you are trying to track down problems, the first step in diagnosing some problems is to notice that messages are not going to the channel you expected or intended, and having logging enabled for all channels can help you investigate such problems.

▼ To Enable MTA Logging on a Specific Channel

1 Edit the `imta.cnf` file.

The file is located in the `/opt/SUNWmsgsr/config` directory.

2 To enable logging for a particular channel, add the `logging` keyword to the channel definition. For example:

*channel-name keyword1 keyword2 **logging***

In addition, you can also set a number of configuration parameters such as directory path for log files, log levels, and so on. See “[25.4 Managing Message Store, Admin, and Default Service Logs](#)” on page 820

Note – The message return job, which runs every night around midnight, appends any existing `mail.log_yesterday` to the cumulative log file, `mail.log`, renames the current `mail.log_current` file to `mail.log_yesterday`, and then begins a new `mail.log_current` file. It also performs the analogous operations for any `connection.log*` files. It is possible that `mail.log_yesterday` contains time stamps which have already passed over rotation time.

▼ To Enable MTA Logging on All Channels

1 Edit the `imta.cnf` file.

The file is located in the `/opt/SUNWmsgsr/config` directory.

- 2 Add the logging keyword to your defaults channel configuration file (see “[12.1 Configuring Channel Defaults](#)” on page 318). For example:

```
defaults logging notices 1 2 4 7 copywarnpost copysendpost postheadonly
noswitchchannel immnonurgent maxjobs 7 defaulthost siroe.com siroe.com

!
! delivery channel to local /var/mail store
l subdirs 20 viaaliasrequired maxjobs 7 pool LOCAL_POOL
mailhost.siroe.com
```

25.3.3 Specifying Additional MTA Logging Options

In addition to the basic information always provided when logging is enabled, you can specify that additional, optional information fields be included by setting various LOG_* MTA options in the MTA Option file. The file specified with the IMTA_OPTION_FILE option in the IMTA tailor file (*msg-svr-base/config/imta_tailor*) specifies the MTA Option file. By default, this is the *msg-svr-base/config/option.dat* file.

For complete details about the MTA Option file, see the “[Option File](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

This section consists of the following subsections:

- “[To Send MTA Logs to syslog](#)” on page 801
- “[To Control Formatting of Log Entries](#)” on page 802
- “[To Correlate Log Message Entries](#)” on page 804
- “[To Log Amount of Time Messages Have Spent in the Queue](#)” on page 804
- “[To Identify Message Delivery Retries](#)” on page 804
- “[To Log TCP/IP Connections](#)” on page 804
- “[To Write Entries to the connection.log File](#)” on page 805
- “[To Correlate Log Messages by Process ID](#)” on page 805
- “[To Save User Names Associated with a Process That Enqueues Mail to the mail.log File](#)” on page 805

▼ To Send MTA Logs to syslog

- 1 Edit the MTA Option file.
- 2 Set the LOG_MESSAGES_SYSLOG option to 1.

A value of 0 disables generation of the syslog notices. A non-zero value enables generation of the syslog notices, with the absolute value controlling the syslog priority and facility mask. (Positive values mean syslog notices and the regular mail.log* entries; negative values, which are not recommended, mean syslog notices only, disabling the regular mail.log* entries. A value of 0 is the default and indicates that syslog (event log) logging is not performed.)

▼ To Control Formatting of Log Entries

1 Edit the MTA option.dat file.

2 Set the LOG_FORMAT option.

- 1 (default) the standard format.
- 2 requests non-null formatting: empty address fields are converted to the string "<>."
- 3 requests counted formatting: all variable length fields are preceded by N, where N is a count of the number of characters in the field.
- 4 causes log entries to be written in an XML-compatible format. Entry log entry appears as a single XML element containing multiple attributes and no sub-elements. Three elements are currently defined, en for enqueue/dequeue entries, co for connection entries, and he for header entries.

Enqueue/dequeue (en) elements can have the following attributes:

```
ts - time stamp (always present)
no - node name (present if LOG_NODE=1)
pi - process id (present if LOG_PROCESS=1)
sc - source channel (always present)
dc - destination channel (always present)
ac - action (always present)
sz - size (always present)
so - source address (always present)
od - original destination address (always present)
de - destination address (always present)
rf - recipient flags (present if LOG_NOTARY=1)
fi - filename (present if LOG_FILENAME=1)
ei - envelope id (present if LOG_ENVELOPE_ID=1)
mi - message id (present if LOG_MESSAGE_ID=1)
us - username (present if LOG_USERNAME=1)
ss - source system (present if bit 0 of LOG_CONNECTION
    is set and source system information is available)
se - sensitivity (present if LOG_SENSITIVITY=1)
pr - priority (present if LOG_PRIORITY=1)
in - intermediate address (present if LOG_INTERMEDIATE=1)
ia - initial address (present if bit 0 of LOG_INTERMEDIATE
    is set and intermediate address information is available)
fl - filter (present if LOG_FILTER=1 and filter information
    is available)
re - reason (present if LOG_REASON=1 and reason string is set)
di - diagnostic (present if diagnostic info available)
tr - transport information (present if bit 5 of LOG_CONNECTION
    is set and transport information is available)
ap - application information (present if bit 6 of LOG_CONNECTION
    is set and application information is available)
```

qt - the amount of time a message has spent in the queue (LOG_QUEUE_TIME=1)

Here is a sample en entry:

```
<en ts="2004-12-08T00:40:26.70" pi="0d3730.10.43" sc="tcp_local"
dc="l" ac="E" sz="12" so="info-E8944AE8D033CB92C2241E@whittlesong.com"
od="rfc822;ned+2Bcharsets@mauve.sun.com"
de="ned+charsets@mauve.sun.com" rf="22"
fi="/path/ZZ01L14XPX0DTM00IKA8.00" ei="01LI4XPQR2EU00IKA8@mauve.sun.com"
mi=<1la3b401c4dd01$7c1clee0$1906fad@elara>" us=""
ss="elara.whittlesong.com ([208.250.6.25])"
in="ned+charsets@mauve.sun.com" ia="ietf-charsets@innosoft.com"
fl="spamfilter1:rvLiXh158xWdQKa9iJ0d7Q==, addheader, keep"/>
```

Note that this entry has been wrapped for clarity; actual log file entries always appear on a single line.

Connection (co) entries can have the following attributes:

- ts - time stamp (always present, also used in en entries)
- no - node name (present if LOG_NODE=1, also used in en entries)
- pi - process id (present if LOG_PROCESS=1, also used in en entries)
- sc - source channel (always present, also used in en entries)
- dr - direction (always present)
- ac - action (always present, also used in en entries)
- tr - transport information (always present, also used in en entries)
- ap - application information (always present, also used in en entries)
- mi - message id (present only if message id info available,
also used in en entries)
- us - username (present only if username information available, also
used in en entries)
- di - diagnostic (present only if diagnostic information available,
also used in en entries)
- ct - the amount of time a message has spent in the queue (LOG_QUEUE_TIME=1,
also used in en entries)

Here is a sample co entry:

```
<co ts="2004-12-08T00:38:28.41" pi="1074b3.61.281" sc="tcp_local" dr="+"
ac="0" tr="TCP|209.55.107.55|25|209.55.107.104|33469" ap="SMTP"/>
```

Header (he) entries have the following attributes:

- ts - time stamp (always present, also used in en entries)
- no - node name (present if LOG_NODE=1, also used in en entries)
- pi - process id (present if LOG_PROCESS=1, also used in en entries)
- va - header line value (always present)

Here is a sample he entry:

```
<he ts="2004-12-08T00:38:31.41" pi="1074b3.61.281" va="Subject: foo"/>
```

▼ To Correlate Log Message Entries

- 1 **Edit the MTA Option file.**
- 2 **Set the LOG_MESSAGE_ID option to 1.**

A value of 0 is the default and indicates that message IDs are not saved in the `mail.log` file.

▼ To Log Amount of Time Messages Have Spent in the Queue

- 1 **Edit the MTA Option file.**
- 2 **Set the LOG_QUEUE_TIME option to 1.**

This option logs the amount of time a message has spent in the queue. The queue time is logged as an integer value in seconds. It appears immediately after the application information string in non-XML format logs. The attribute name in XML formatted logs for this value is `qt`.

▼ To Identify Message Delivery Retries

- 1 **Edit the MTA Option file.**
- 2 **Set the LOG_FILENAME option to 1.**

This option makes it easier to immediately spot how many times the delivery of a particular message file has been retried. This option can also be useful in understanding when the MTA does or does not split a message to multiple recipients into separate message file copies on disk.

▼ To Log TCP/IP Connections

- 1 **Edit the MTA Option file.**
- 2 **Set the LOG_CONNECTION option.**

This option causes the MTA to log TCP/IP connections, as well as message traffic. The connection log entries are written to the `mail.log*` files by default. Optionally, the connection log entries can be written to `connection.log*` files. See the `SEPARATE_CONNECTION_LOG` option for more information.

▼ To Write Entries to the `connection.log` File

- 1 Edit the MTA Option file.
- 2 Set the `SEPARATE_CONNECTION_LOG` option to 1.

Use this option to specify that connection log entries instead be written to `connection.log` files. The default value of 0 causes the connection logging to be stored in the MTA log files.

▼ To Correlate Log Messages by Process ID

- 1 Edit the MTA Option file.
- 2 Set the `LOG_PROCESS` option.

When used in conjunction with `LOG_CONNECTION`, this option enables correlation by process ID of which connection entries correspond to which message entries.

▼ To Save User Names Associated with a Process That Enqueues Mail to the `mail.log` File

- 1 Edit the MTA Option file.
- 2 Set the `LOG_USERNAME` option.

This option controls whether or not the user name associated with a process that enqueues mail is saved in the `mail.log` file. For SMTP submissions where SASL (SMTP AUTH) is used, the user name field will be the authenticated user name (prefixed with an asterisk character).

25.3.4 MTA Message Logging Examples

The exact field format and list of fields logged in the MTA message files vary according to the logging options set. This section shows a few examples of interpreting typical sorts of log entries. This section consists of the following subsections:

- “25.3.4.1 MTA Logging Example: User Sends an Outgoing Message” on page 806
- “25.3.4.2 MTA Logging Example: Including Optional Logging Fields” on page 807
- “25.3.4.3 MTA Logging Example – Sending to a List” on page 808
- “25.3.4.4 MTA Logging – Sending to a Nonexistent Domain” on page 809
- “25.3.4.5 MTA Logging Example – Sending to a Nonexistent Remote User” on page 810
- “25.3.4.6 MTA Logging Example – Rejecting a Remote Side’s Attempt to Submit a Message” on page 812
- “25.3.4.7 MTA Logging Example – Multiple Delivery Attempts” on page 813
- “25.3.4.8 MTA Logging – Incoming SMTP Message Routed Through the Conversion Channel” on page 814

- “25.3.4.9 MTA Logging Example: Outbound Connection Logging” on page 815
- “25.3.4.10 MTA Logging Example: Inbound Connection Logging” on page 817

For a description of additional, optional fields, see “25.3.3 Specifying Additional MTA Logging Options” on page 801.

Note – For typographic reasons, log file entries will be shown folded onto multiple lines—actual log file entries are one line per entry.

When reviewing a log file, keep in mind that on a typical system many messages are being handled at once. Typically, the entries relating to a particular message will be interspersed among entries relating to other messages being processed during that same time. The basic logging information is suitable for gathering a sense of the overall numbers of messages moving through the MTA.

If you wish to correlate particular entries relating to the same message to the same recipient(s), enable LOG_MESSAGE_ID. To correlate particular messages with particular files in the MTA queue area, or to see from the entries how many times a particular not-yet-successfully-dequeued message has had delivery attempted, enable LOG_FILENAME. For SMTP messages (handled via a TCP/IP channel), if you want to correlate TCP connections to and from remote systems with the messages sent, enable LOG_PROCESS and some level of LOG_CONNECTION.

25.3.4.1

MTA Logging Example: User Sends an Outgoing Message

The example below shows a fairly basic example of the sort of log entries one might see if a local user sends a message out an outgoing TCP/IP channel, for example, to the Internet. In this example, LOG_CONNECTION is enabled. The lines marked with (1) and (2) are one entry—they would appear on one physical line in an actual log file. Similarly, the lines marked with (3) - (7) are one entry and would appear on one physical line.

EXAMPLE 25-1 MTA Logging: A Local User Sends An Outgoing Message

```
16-Feb-2007 15:41:32.36 tcp_intranet tcp_local    EE 1      (1)
adam@sesta.com rfc822;marlowe@siroe.com marlowe@siroe.com      (2)
siroe.com (siroe.com [192.160.253.66])  

  
16-Feb-2007 15:41:34.73 tcp_local              DE 1      (3)
adam@sesta.com rfc822;marlowe@siroe.com marlowe@siroe.com      (4)
thor.siroe.com dns;thor.siroe.com  

  
(TCP|206.184.139.12|2788|192.160.253.66|25)           (5)  

  
(thor.siroe.com ESMTP Sendmail ready Thu 15 Feb 2007 21:37:29 -0700 [MST])   (6)  

  
smtp;250 2.1.5 <marlowe@siroe.com>... Receipt ok          (7)
```

EXAMPLE 25-1 MTA Logging: A Local User Sends An Outgoing Message *(Continued)*

1. This line shows the date and time of an enqueue with ESMTP (EE) from the `tcp_intranet` channel to the `tcp_local` channel of a one (1) block message.
2. This is part of the same physical line of the log file as (1), presented here as a separate line for typographical convenience. It shows the envelope `From:` address, in this case `adam@sesta.com`, and the original version and current version of the envelope `To:` address, in this case `marlowe@siroe.com`.
3. This shows the date and time of a dequeue with ESMTP (DE) from the `tcp_local` channel of a one (1) block message that is, a successful send by the `tcp_local` channel to some remote SMTP server.
4. This shows the envelope `From:` address, the original envelope `To:` address, and the current form of the envelope `To:` address.
5. This shows that the actual system to which the connection was made is named `thor.siroe.com` in the DNS, that the local sending system has IP address `206.184.139.12` and is sending from port `2788`, that the remote destination system has IP address `192.160.253.66` and the connection port on the remote destination system is port `25`.
6. This shows the SMTP banner line of the remote SMTP server.
7. This shows the SMTP status code returned for this address; `250` is the basic SMTP success code and in addition, this remote SMTP server responds with extended SMTP status codes and some additional text.

25.3.4.2**MTA Logging Example: Including Optional Logging Fields**

This example shows a logging entry similar to that shown in [Example 25-3](#) with `LOG_FILENAME=1` and `LOG_MESSAGE_ID=1` showing the file name (1 and 3 below) and message ID (2 and 4 below). The message ID in particular can be used to correlate which entries relate to which message.

EXAMPLE 25-2 MTA Logging – Including Optional Logging Fields

```
16-Feb-2007 15:41:32.36 tcp_intranet tcp_local      EE 1
adaml@sesta.com rfc822;marlowe@siroe.com marlowe@siroe.com
/opt/SUNWmsgsr/data/queue/tcp_local/002/ZZf0r4i0Wdy51.01      (1)
<0JDJ00D02IBwdx00@sesta.com>                            (2)
siroe.com (siroe.com [192.160.253.66])

16-Feb-2007 15:41:34.73 tcp_local                  DE 1
adaml@sesta.com rfc822;marlowe@siroe.com marlowe@siroe.com
/opt/SUNWmsgsr/data/queue/tcp_local/002/ZZf0r4i0Wdy51.01      (3)
<0JDJ00D02IBwdx00@sesta.com>                            (4)
thor.siroe.com dns;thor.siroe.com
(TCP|206.184.139.12|2788|192.160.253.66|25)
(siroe.com ESMTP Sendmail ready at Thu, 15 Feb 2007 21:37:29 -0700 [MST])
```

EXAMPLE 25-2 MTA Logging – Including Optional Logging Fields *(Continued)*

```
smtp;250 2.1.5 <marlowe@siroe.com>... Recipient ok
```

25.3.4.3

MTA Logging Example – Sending to a List

This example illustrates sending to multiple recipients with LOG_FILENAME=1, LOG_MESSAGE_ID=1, and LOG_CONNECTION=1 enabled. Here user adam@sesta.com has sent to the MTA mailing list test-list@sesta.com, which expanded to bob@sesta.com, carol@varrius.com, and david@varrius.com. Note that the original envelope To: address is test-list@sesta.com for each recipient, though the current envelope To: address is each respective address. Note how the message ID is the same throughout, though two separate files (one for the l channel and one going out the tcp_local channel) are involved.

EXAMPLE 25-3 MTA Logging – Sending to a List

```
20-Feb-2007 14:00:16.46 tcp_local      tcp_local      EE 1
adam@sesta.com rfc822;test-list@sesta.com carol@varrius.com
/opt/SUNWmsgsr/data/queue/tcp_local/004/ZZf0r2D0yuej4.01
<0JDQ00706R0FX100@sesta.com>
siroe.com (siroe.com [192.160.253.66])

20-Feb-2007 14:00:16.47 tcp_local      tcp_local      EE 1
adam@sesta.com rfc822;test-list@sesta.com david@varrius.com
/opt/SUNWmsgsr/data/queue/tcp_local/004/ZZf0r2D0yuej4.01
<0JDQ00706R0FX100@sesta.com>
siroe.com (siroe.com [192.160.253.66])

20-Feb-2007 14:00:16.48 tcp_local      ims-ms        EE 1
adam@sesta.com rfc822;test-list@sesta.com bob@ims-ms-daemon
/opt/SUNWmsgsr/data/queue/ims-ms/008/ZZf0r2D0yuej6.01
<0JDQ00706R0FX100@sesta.com>
siroe.com (siroe.com [192.160.253.66])

20-Feb-2007 14:00:16.68 ims-ms          D 1
adam@sesta.com rfc822;test-list@sesta.com bob@ims-ms-daemon
/opt/SUNWmsgsr/data/queue/ims-ms/008/ZZf0r2D0yuej6.01
<0JDQ00706R0FX100@sesta.com>

20-Feb-2007 14:00:17.73 tcp_local      DE 1
adam@sesta.com rfc822;test-list@sesta.com carol@varrius.com
/opt/SUNWmsgsr/data/queue/tcp_local/004/ZZf0r2D0yuej4.01
<0JDQ00706R0FX100@sesta.com>
gw.varrius.com dns;gw.varrius.com (TCP|206.184.139.12|2788|192.160.253.66|25)
(gw.varrius.com -- SMTP Sendmail)
smtp;250 2.1.5 <carol@varrius.com>... Recipient ok
```

EXAMPLE 25-3 MTA Logging – Sending to a List *(Continued)*

```
20-Feb-2007 14:00:17.75 tcp_local DE 1
adam@sesta.com rfc822;test-list@sesta.com david@varrius.com
/opt/SUNWmsgsr/data/queue/tcp_local/004/ZZf0r2D0yuej4.01
<0JDQ00706R0FX100@sesta.com>
gw.varrius.com dns;gw.varrius.com (TCP|206.184.139.12|2788|192.160.253.66|25)
(gw.varrius.com -- SMTP Sendmail)
smtp;250 2.1.5 <david@varrius.com>... Recipient ok
```

25.3.4.4**MTA Logging – Sending to a Nonexistent Domain**

This example illustrates an attempt to send to a nonexistent domain (here `very.bogus.com`); that is, sending to a domain name that is not noticed as nonexistent by the MTA's rewrite rules and that the MTA matches to an outgoing TCP/IP channel. This example assumes the MTA option settings of `LOG_FILENAME=1` and `LOG_MESSAGE_ID=1`.

When the TCP/IP channel runs and checks for the domain name in the DNS, the DNS returns an error that no such name exists. Note the “rejection” entry (5), with the DNS returning an error that this is not a legal domain name, as seen in (6).

Because the address is rejected after the message has been submitted, the MTA generates a bounce message to the original sender. The MTA enqueues the new rejection message to the original sender (1), and sends a copy to the postmaster (4) before deleting the original outbound message (the R entry shown in (5)).

Notification messages, such as bounce messages, have an empty envelope `From:` address—as seen, for instance, in (2) and (8)—in which the envelope `From:` field is shown as an empty space. The initial enqueue of a bounce message generated by the MTA shows the message ID for the new notification message followed by the message ID for the original message (3). (Such information is not always available to the MTA, but when it is available to be logged, it allows correlation of the log entries corresponding to the outbound failed message with the log entries corresponding to the resulting notification message.) Such notification messages are enqueued to the process channel, which in turn enqueues them to an appropriate destination channel (7).

EXAMPLE 25-4 MTA Logging – Sending to a Nonexistent Domain

```
20-Feb-2007 14:17:07.77 tcp_intranet tcp_local E 1
adam@sesta.com rfc822;user@very.bogus.com user@very.bogus.com
/opt/SUNWmsgsr/data/queue/tcp_local/008/ZZf0r2D0CVaL0.00
<0JDQ00903RS89T00@sesta.com>
siroe.com (siroe.com [192.160.253.66])

20-Feb-2007 14:17:08.24 tcp_local process E 1 (1)
rfc822;adam@sesta.com adam@sesta.com (2)
/opt/SUNWmsgsr/data/queue/process/ZZf0r2D0CVbR0.00
<0JDQ00904RSK9Z00@sesta.com>,<0JDQ00903RS89T00@sesta.com> (3)
```

EXAMPLE 25–4 MTA Logging – Sending to a Nonexistent Domain *(Continued)*

```
tcp-daemon.mailhost.sesta.com

20-Feb-2007 14:17:08.46 tcp_local      process      E 1      (4)
rfc822;postmaster@sesta.com postmaster@sesta.com
/opt/SUNWmsgsr/data/queue/process/ZZf0r2D0CVbR1.00
<0JDQ00906RSK9Z00@sesta.com>,<0JDQ00903RS89T00@sesta.com>
tcp-daemon.mailhost.sesta.com

20-Feb-2007 14:17:08.46 tcp_local          R 1      (5)
adam@sesta.com rfc822;user@very.bogus.com user@very.bogus.com
/opt/SUNWmsgsr/data/queue/tcp_local/008/ZZf0r2D0CVaL0.00
<0JDQ00903RS89T00@sesta.com>
Illegal host/domain name found          (6)
(TCP active open: Failed gethostbyname() on very.bogus.com, resolver errno = 1)

20-Feb-2007 14:17:09.21 process      ims-ms      E 3      (7)
rfc822;adam@sesta.com adam@ims-ms-daemon          (8)
/opt/SUNWmsgsr/data/queue/ims-ms/018/ZZf0r2D0CVbS1.00
<0JDQ00904RSK9Z00@sesta.com>
process-daemon.mailhost.sesta.com

20-Feb-2007 14:17:09.72 process      ims-ms      E 3
rfc822;postmaster@sesta.com postmaster@ims-ms-daemon
/opt/SUNWmsgsr/data/queue/ims-ms/014/ZZf0r2D0CVbS2.00
<0JDQ00906RSK9Z00@sesta.com>
process-daemon.mailhost.sesta.com

20-Feb-2007 14:17:09.73 ims-ms          D 3
rfc822;adam@sesta.com adam@ims-ms-daemon
/opt/SUNWmsgsr/data/queue/ims-ms/018/ZZf0r2D0CVbS1.00
<0JDQ00904RSK9Z00@sesta.com>

20-Feb-2007 14:17:09.84 ims-ms          D 3
rfc822;postmaster@sesta.com postmaster@ims-ms-daemon
/opt/SUNWmsgsr/data/queue/ims-ms/014/ZZf0r2D0CVbS2.00
<0JDQ00906RSK9Z00@sesta.com>
```

25.3.4.5**MTA Logging Example – Sending to a Nonexistent Remote User**

This example illustrates an attempt to send to a bad address on a remote system. This example assumes MTA option settings of `LOG_FILENAME=1` and `LOG_MESSAGE_ID=1`, and channel option settings of `LOG_BANNER=1` and `LOG_TRANSPORTINFO=1`. Note the rejection entry (R), seen in (1). But in contrast to the rejection entry in [Example 25–4](#), note that the rejection entry here shows that a connection to a remote system was made, and shows the SMTP error code issued by the remote SMTP server, (2) and (3). The inclusion of the information shown in (2) is due to setting the channel options `LOG_BANNER=1` and `LOG_TRANSPORTINFO=1`.

EXAMPLE 25-5 MTA Logging – Sending to a Nonexistent Remote User

```

26-Feb-2007 13:56:35.16 tcp_intranet tcp_local    EE 1
adam@sesta.com rfc822;nonesuch@siroe.com nonesuch@siroe.com
/opt/SUNWmsgsr/data/queue/tcp_local/000/ZZf0s690a3mf2.01
<0JE100J08UU24H00@sesta.com>
siroe.com (siroe.com [192.160.253.66])

26-Feb-2007 13:56:35.19 tcp_local    process      E 1
rfc822;adam@sesta.com adam@sesta.com
/opt/SUNWmsgsr/data/queue/process/ZZf0s690a3ml2.00
<0JE100J09UUB4N00@sesta.com>,<0JE100J08UU24H00@sesta.com>
tcp-daemon.mailhost.sesta.com

26-Feb-2007 13:56:35.20 tcp_local    process      E 1
rfc822;postmaster@sesta.com postmaster@sesta.com
/opt/SUNWmsgsr/data/queue/process/ZZf0s690a3ml3.00
<0JE100J0BUUB4N00@sesta.com>,<0JE100J08UU24H00@sesta.com>
tcp-daemon.mailhost.sesta.com

26-Feb-2007 13:56:35.20 tcp_local          RE 1      (1)
adam@sesta.com rfc822;nonesuch@siroe.com nonesuch@siroe.com
/opt/SUNWmsgsr/data/queue/tcp_local/000/ZZf0s690a3mf2.01
<0JE100J08UU24H00@sesta.com>
thor.siroe.com dns;thor.siroe.com
(TCP|206.184.139.12|2788|192.160.253.66|25)           (2)
(thor.siroe.com -- Server ESMTP [Sun Java System Messaging
Server 6.2-8.01 [built Feb 16 2007]])
smtp;550 5.1.1 unknown or illegal alias: nonesuch@siroe.com      (3)

26-Feb-2007 13:56:35.62 process      ims-ms      E 4
rfc822;adam@sesta.com adam@ims-ms-daemon
/opt/SUNWmsgsr/data/queue/ims-ms/003/ZZf0s690a3mm5.00
<0JE100J09UUB4N00@sesta.com>
process-daemon.mailhost.sesta.com

26-Feb-2007 13:56:36.07 process      ims-ms      E 4
rfc822;postmaster@sesta.com postmaster@ims-ms-daemon
/opt/SUNWmsgsr/data/queue/ims-ms/016/ZZf0s690a3nm7.01
<0JE100J0BUUB4N00@sesta.com>
process-daemon.mailhost.sesta.com

26-Feb-2007 13:56:35.83 ims-ms        D 4
rfc822;adam@sesta.com adam@ims-ms-daemon
/opt/SUNWmsgsr/data/queue/ims-ms/003/ZZf0s690a3mm5.00
<0JE100J09UUB4N00@sesta.com>

26-Feb-2007 13:56:36.08 ims-ms        D 4

```

EXAMPLE 25-5 MTA Logging – Sending to a Nonexistent Remote User *(Continued)*

```
rfc822;postmaster@sesta.com postmaster@ims-ms-daemon
/opt/SUNWmsgsr/data/queue/ims-ms/016/ZZf0s690a3nm7.01
<0JE100J0BUUB4N00@sesta.com>
```

25.3.4.6

MTA Logging Example – Rejecting a Remote Side's Attempt to Submit a Message

This example illustrates the sort of log file entry resulting when the MTA rejects a remote side's attempt to submit a message. (This example assumes that no optional LOG_* options are enabled, so only the basic fields are logged in the entry. Note that enabling the LOG_CONNECTION option, in particular, would result in additional informative fields in such J entries.) In this case, the example is for an MTA that has set up SMTP relay blocking (see “[18.7 Configuring SMTP Relay Blocking](#)” on page 560) with an ORIG_SEND_ACCESS mapping, including:

```
ORIG_SEND_ACCESS

! ...numerous entries omitted...
!
tcp_local|*|tcp_local|* $NRelaying$ not$ permitted
```

and where alan@very.bogus.com is not an internal address. Hence the attempt of the remote user harold@varrius.com to relay through the MTA system to the remote user alan@very.bogus.com is rejected.

EXAMPLE 25-6 MTA Logging – Rejecting a Remote Side's Attempt to Submit a Message

```
26-Feb-2007 14:10:06.89 tcp_local          JE 0  (1)
harold@varrius.com rfc822; alan@very.bogus.com      (2)
530 5.7.1 Relaying not allowed: alan@very.bogus.com (3)
```

1. This log shows the date and time the MTA rejects a remote side's attempt to submit a message. The rejection is indicated by a J record. (Cases where an MTA channel is attempting to send a message which is rejected is indicated by R records, as shown in [Example 25-4](#) and [Example 25-5](#)).

Note – The last J record written to the log will have an indication stating that it is the last for a given session. Also, the current version of Messaging Server does not place a limit on the number of J records.

2. The attempted envelope From: and To: addresses are shown. In this case, no original envelope To: information was available so that field is empty.
3. The entry includes the SMTP error message the MTA issued to the remote (attempted sender) side.

25.3.4.7

MTA Logging Example – Multiple Delivery Attempts

This example illustrates the sort of log file entries resulting when a message cannot be delivered upon the first attempt, so the MTA attempts to send the message several times. This example assumes option settings of `LOG_FILENAME=1` and `LOG_MESSAGE_ID=1`.

EXAMPLE 25-7 MTA Logging – Multiple Delivery Attempts

```

26-Feb-2007 14:38:16.27 tcp_intranet tcp_local    EE 1          (1)
adams@sesta.com rfc822;user@some.org user@some.org
/opt/SUNWmsgsr/data/queue/tcp_local/001/ZZf0s690kN_y0.00
<0JE100L05WRJIC00@sesta.com>

26-Feb-2007 14:38:16.70 tcp_local             Q 1          (2)
adams@sesta.com rfc822;user@some.org user@some.org
/opt/SUNWmsgsr/data/queue/tcp_local/001/ZZf0s690kN_y0.00      (3)
<0JE100L05WRJIC00@sesta.com>
TCP active open: Failed connect() 192.1.1.1:25  Error: no route to host (4)

...several hours worth of entries...

26-Feb-2007 16:58:11.20 tcp_local             Q 1          (5)
adams@sesta.com rfc822;user@some.org user@some.org
/opt/SUNWmsgsr/data/queue/tcp_local/001/ZYf0s690kN_y0.01      (6)
<0JE100L05WRJIC00@sesta.com>
TCP active open: Failed connect() 192.1.1.1:25  Error: no route to host

...several hours worth of entries...

26-Feb-2007 19:15:12.11 tcp_local             Q 1          (7)
adams@sesta.com rfc822;user@some.org user@some.org
/opt/SUNWmsgsr/data/queue/tcp_local/001/ZXf0s690kN_y0.00
<0JE100L05WRJIC00@sesta.com>
TCP active open: Failed connect() 192.1.1.1:25  Error: Connection refused (8)

...several hours worth of entries...

26-Feb-2007 22:41:12.63 tcp_local             DE 1          (9)
adams@sesta.com rfc822;user@some.org user@some.org
/opt/SUNWmsgsr/data/queue/tcp_local/001/ZXf0s690kN_y0.00
<0JE100L05WRJIC00@sesta.com>
host.some.org dns;host.some.org (TCP|206.184.139.12|2788|192.1.1.1|25)
(All set, fire away)
smtp;250 2.1.5 <user@some.org>... Recipient ok

```

1. The message comes in the `tcp_internal` channel—perhaps from a POP or IMAP client, or perhaps from another host within the organization using the MTA as an SMTP relay; the MTA enqueues it to the outgoing `tcp_local` channel.
2. The first delivery attempt fails, as indicated by the `Q` entry.

EXAMPLE 25-7 MTA Logging – Multiple Delivery Attempts *(Continued)*

3. That this is a first delivery attempt can be seen from the ZZ* filename.
4. This delivery attempt failed when the TCP/IP package could not find a route to the remote side. As opposed to [Example 25-4](#), the DNS did not object to the destination domain name, some.org; rather, the “no route to host” error indicates that there is some network problem between the sending and receiving side.
5. The next time the MTA periodic job runs it reattempts delivery, again unsuccessfully.
6. The file name is ZY*, indicating that this is a second attempt.
7. The file name is ZX* for this third unsuccessful attempt.
8. The next time the periodic job reattempts delivery the delivery fails, though this time the TCP/IP package is not complaining that it cannot get through to the remote SMTP server, but rather the remote SMTP server is not accepting connections. (Perhaps the remote side fixed their network problem, but has not yet brought their SMTP server back up—or their SMTP server is swamped handling other messages and hence was not accepting connections at the moment the MTA tried to connect.)
9. Finally the message is dequeued.

25.3.4.8**MTA Logging – Incoming SMTP Message Routed Through the Conversion Channel**

This example illustrates the case of a message routed through the conversion channel. The site is assumed to have a CONVERSIONS mapping table such as:

```
CONVERSIONS
IN-CHAN=tcp_local;OUT-CHAN=ims-ms;CONVERT Yes
```

This example assumes option settings of LOG_FILENAME=1 and LOG_MESSAGE_ID=1.

EXAMPLE 25-8 MTA Logging – Incoming SMTP Message Routed Through the Conversion Channel

```
26-Feb-2007 15:31:04.17 tcp_local conversion EE 1 (1)
amy@siroe.edu rfc822;bert@sesta.com bert@ims-ms-daemon
/opt/SUNWmsgsr/data/queue/conversion/ZZf0s090wFwx2.01
<0JE100206Z7J5F00@siroe.edu>
```

```
26-Feb-2007 15:31:04.73 conversion ims-ms E 1 (2)
amy@siroe.edu rfc822;bert@sesta.com bert@ims-ms-daemon
/opt/SUNWmsgsr/data/queue/ims-ms/007/ZZf0s090wMwq1.00
<0JE100206Z7J5F00@siroe.edu>
```

```
26-Feb-2007 15:31:04.73 conversion D 1 (3)
amy@siroe.edu rfc822;bert@sesta.com bert@ims-ms-daemon
/opt/SUNWmsgsr/data/queue/conversion/ZZf0s090wFwx2.01
```

EXAMPLE 25-8 MTA Logging – Incoming SMTP Message Routed Through the Conversion Channel
(Continued)

```
<0JE100206Z7J5F00@siroe.edu>

26-Feb-2007 15:31:04.73 ims-ms D 1 (4)
amy@siroe.edu rfc822;bert@sesta.com bert@ims-ms-daemon
/opt/SUNWmsgsr/data/queue/ims-ms/007/ZZf0s090wMwq1.00
<0JE100206Z7J5F00@siroe.edu>
```

1. The message from external user amy@siroe.edu comes in addressed to the ims-ms channel recipient bert@sesta.com. The CONVERSIONS mapping entry, however, causes the message to be initially enqueued to the conversion channel (rather than directly to the ims-ms channel).
2. The conversion channel runs and enqueues the message to the ims-ms channel.
3. Then the conversion channel can dequeue the message (delete the old message file).
4. And finally the ims-ms channel dequeues (delivers) the message.

25.3.4.9

MTA Logging Example: Outbound Connection Logging

This example illustrates log output for an outgoing message when connection logging is enabled, via LOG_CONNECTION=3, LOG_PROCESS=1, LOG_MESSAGE_ID=1 and LOG_FILENAME=1 are also assumed in this example. The example shows the case of user adam@sesta.com sending the same message (note that the message ID is the same for each message copy) to three recipients, bobby@hosta.sesta.com, carl@hosta.sesta.com, and dave@hostb.sesta.com. This example assumes that the message is going out a tcp_local channel marked (as such channels usually are) with the single_sys channel keyword. Therefore, a separate message file on disk will be created for each set of recipients to a separate host name, as seen in (1), (2), and (3), where the bobby@hosta.sesta.com and carl@hosta.sesta.com recipients are stored in the same message file, but the dave@hostb.sesta.com recipient is stored in a different message file.

EXAMPLE 25-9 MTA Logging: Outbound Connection Logging

```
28-Feb-2007 09:13:19.18 409f.3.1 tcp_intranet tcp_local EE 1
adam@sesta.com rfc822;bobby@hosta.sesta.com bobby@hosta.sesta.com
/opt/SUNWmsgsr/data/queue/tcp_local/000/ZZf0s4g0G2Zt0.00 (1)
<0JE500C0371HRJ00@sesta.com>
siroe.com (siroe.com [192.160.253.66])
```

```
28-Feb-2007 09:13:19.18 409f.3.1 tcp_intranet tcp_local EE 1
adam@sesta.com rfc822;carl@hosta.sesta.com carl@hosta.sesta.com
/opt/SUNWmsgsr/data/queue/tcp_local/000/ZZf0s4g0G2Zt0.00 (2)
<0JE500C0371HRJ00@sesta.com>
siroe.com (siroe.com [192.160.253.66])
```

```
28-Feb-2007 09:13:19.19 409f.3.2 tcp_intranet tcp_local EE 1
adam@sesta.com rfc822;dave@hostb.sesta.com dave@hostb.sesta.com
```

EXAMPLE 25–9 MTA Logging: Outbound Connection Logging (Continued)

```

/opt/SUNWmsgsr/data/queue/tcp_local/004/ZZf0s4g0G2Zt1.00          (3)
<0JE500C0371HRJ00@sesta.com>
siroe.com (siroe.com [192.160.253.66])

28-Feb-2007 09:13:19.87 40a5.2.0 tcp_local - 0          (4)
TCP|206.184.139.12|5900|206.184.139.66|25
SMTP/hostb.sesta.com/mailhub.sesta.com          (5)

28-Feb-2007 09:13:20.23 40a5.3.4 tcp_local - 0          (6)
TCP|206.184.139.12|5901|206.184.139.70|25
SMTP/hosta.sesta.com/hosta.sesta.com          (7)

28-Feb-2007 09:13:20.50 40a5.2.5 tcp_local DE 1
adam@sesta.com rfc822;bobby@hosta.sesta.com bobby@hosta.sesta.com
/opt/SUNWmsgsr/data/queue/tcp_local/000/ZZf0s4g0G2Zt0.00
<0JE500C0371HRJ00@sesta.com>
hosta.sesta.com dns;hosta.sesta.com          (8)
(TCP|206.184.139.12|5901|206.184.139.70|25)
(hosta.sesta.com -- Server ESMTP [Sun Java System Messaging Server
6.2-8.01 [built Feb 16 2007]])
smtp;250 2.1.5 bobby@hosta.sesta.com and options OK.

28-Feb-2007 09:13:20.50 40a5.2.5 tcp_local DE 1
adam@sesta.com rfc822;carl@hosta.sesta.com carl@hosta.sesta.com
/opt/SUNWmsgsr/data/queue/tcp_local/000/ZZf0s4g0G2Zt0.00
<0JE500C0371HRJ00@sesta.com>
hosta.sesta.com dns;hosta.sesta.com
(TCP|206.184.139.12|5901|206.184.139.70|25)
(hosta.sesta.com -- Server ESMTP [Sun Java System Messaging Server
6.2-8.01 [built Feb 16 2007]])
smtp;250 2.1.5 carl@hosta.sesta.com and options OK.

28-Feb-2007 09:13:20.50 40a5.2.6 tcp_local - C          (9)
TCP|206.184.139.12|5901|206.184.139.70|25
SMTP/hosta.sesta.com/hosta.sesta.com

28-Feb-2007 09:13:21.13 40a5.3.7 tcp_local DE 1
adam@sesta.com rfc822;dave@hostb.sesta.com dave@hostb.sesta.com
/opt/SUNWmsgsr/data/queue/tcp_local/004/ZZf0s4g0G2Zt1.00
<0JE500C0371HRJ00@sesta.com>
mailhub.sesta.com dns;mailhub.sesta.com
(TCP|206.184.139.12|5900|206.184.139.66|25)
(mailhub.sesta.com ESMTP Sendmail ready at Tue, 27 Feb 2007 22:19:40 GMT)
smtp;250 2.1.5 <dave@hostb.sesta.com>... Recipient ok

28-Feb-2007 09:13:21.33 40a5.3.8 tcp_local - C          (10)

```

EXAMPLE 25-9 MTA Logging: Outbound Connection Logging (Continued)

TCP|206.184.139.12|5900|206.184.139.66|25
SMTP/hostb.sesta.com/mailhub.sesta.com

1. The message is enqueued to the first recipient...
2.and to the second recipient...
3.and to the third recipient.
4. Having LOG_CONNECTION=3 set causes the MTA to write this entry. The minus, -, indicates that this entry refers to an outgoing connection. The 0 means that this entry corresponds to the opening of the connection. Also note that the process ID here is the same, 40a5, since the same process is used for the multithreaded TCP/IP channel for these separate connection opens, though this open is being performed by thread 2 vs. thread 3.
5. As there are two separate remote systems to which to connect, the multithreaded SMTP client in separate threads opens up a connection to each—the first in this entry, and the second shown in 7. This part of the entry shows the sending and destination IP numbers and port numbers, and shows both the initial host name, and the host name found by doing a DNS lookup. In the *SMTP/initial-host/dns-host* clauses, note the display of both the initial host name, and that used after performing a DNS MX record lookup on the initial host name: *mailhub.sesta.com* is apparently an MX server for *hostb.sesta.com*.
6. The multithreaded SMTP client opens up a connection to the second system in a separate thread (though the same process).
7. As there are two separate remote systems to which to connect, the multithreaded SMTP client in separate threads opens up a connection to each—the second in this entry, and the first shown above in 5. This part of the entry shows the sending and destination IP numbers and port numbers, and shows both the initial host name, and the host name found by doing a DNS lookup. In this example, the system *hosta.sesta.com* apparently receives mail directly itself.
8. Besides resulting in specific connection entries, LOG_CONNECTION=3 also causes inclusion of connection related information in the regular message entries, as seen here for instance.
9. Having LOG_CONNECTION=3 causes the MTA to write this entry. After any messages are dequeued, (the bobby and carl messages in this example), the connection is closed, as indicated by the C in this entry.
10. The connection *mailhub.sesta.com* is closed now that the delivery of the message (dave in this example) is complete.

25.3.4.10**MTA Logging Example: Inbound Connection Logging**

This example illustrates log output for an incoming SMTP message when connection logging is enabled, via LOG_CONNECTION=3.

EXAMPLE 25-10 MTA Logging – Inbound Connection Logging

```
28-Feb-2007 11:50:59.10 tcp_local + 0          (1)
TCP|206.184.139.12|25|192.160.253.66|1244 SMTP      (2)

28-Feb-2007 11:51:15.12 tcp_local    ims-ms    EE 1
service@siroe.com rfc822;adam@sesta.com adam@ims-ms-daemon
THOR.SIROE.COM (THOR.SIROE.COM [192.160.253.66])      (3)

28-Feb-2007 11:51:15.32 ims-ms          D 1
service@siroe.com rfc822;adam@sesta.com adam@ims-ms-daemon

28-Feb-2007 11:51:15.66 tcp_local + C          (4)
TCP|206.184.139.12|25|192.160.253.66|1244 SMTP
```

1. The remote system opens a connection. The 0 character indicates that this entry regards the opening of a connection; the + character indicates that this entry regards an incoming connection.
2. The IP numbers and ports for the connection are shown. In this entry, the receiving system (the system making the log file entry) has IP address 206.184.139.12 and the connection is being made to port 25; the sending system has IP address 192.160.253.66 and is sending from port 1244.
3. In the entry for the enqueue of the message from the incoming TCP/IP channel (`tcp_local`) to the `ims-ms` channel recipient, note that information beyond the default is included since `LOG_CONNECTION=3` is enabled. Specifically, the name that the sending system claimed on its HELO or EHLO line, the sending system's name as found by a DNS reverse lookup on the connection IP number, and the sending system's IP address are all logged; see [Chapter 12, “Configuring Channel Definitions,”](#) behavior.
4. The inbound connection is closed. The C character indicates that this entry regards the closing of a connection; the + character indicates that this entry regards an incoming connection.

25.3.5

Enabling Dispatcher Debugging

Dispatcher error and debugging output (if enabled) are written to the file `dispatcher.log` in the MTA log directory. The dispatcher configuration information is specified in the `msg-svr-base/config/dispatcher.cnf` file. A default configuration file is created at installation time and can be used without any changes made. However, if you want to modify the default configuration file for security or performance reasons, you can do so by editing the `dispatcher.cnf` file.

TABLE 25-5 Dispatcher Debugging Bits

Bit	Usage		
	Hexadecimal value	Decimal value	
0	x 00001	1	Basic Service Dispatcher main module debugging.
1	x 00002	2	Extra Service Dispatcher main module debugging.
2	x 00004	4	Service Dispatcher configuration file logging.
3	x 00008	8	Basic Service Dispatcher miscellaneous debugging.
4	x 00010	16	Basic service debugging.
5	x 00020	32	Extra service debugging.
6	x 00040	64	Process related service debugging.
7	x 00080	128	Not used.
8	x 00100	256	Basic Service Dispatcher and process communication debugging.
9	x 00200	512	Extra Service Dispatcher and process communication debugging.
10	x 00400	1024	Packet level communication debugging.
11	x 00800	2048	Not used.
12	x 01000	4096	Basic Worker Process debugging.
13	x 02000	8192	Extra Worker Process debugging.
14	x 04000	16384	Additional Worker Process debugging, particularly connection hand-offs.
15	x 08000	32768	Not used.
16	x 10000	65536	Basic Worker Process to Service Dispatcher I/O debugging.
17	x 20000	131072	Extra Worker Process to Service Dispatcher I/O debugging.
20	x 100000	1048576	Basic statistics debugging.
21	x 200000	2097152	Extra statistics debugging.
24	x 1000000	16777216	Log PORT_ACCESS denials to the dispatcher.log file.

▼ To Enable Dispatcher Error Debugging Output

1 Edit the `dispatcher.cnf` file.

2 Set the `DEBUG` option to `-1`.

You can also set the logical or environmental variable `IMTA_DISPATCHER_DEBUG` (UNIX), which defines a 32-bit debug mask in hexadecimal, to the value `FFFFFF`. The table above describes the meaning of each bit.

▼ To Set Dispatcher Parameters (Solaris)

The dispatcher services offered in the dispatcher configuration file affects requirements for various system parameters. The system's heap size (`datasize`) must be enough to accommodate the dispatcher's thread stack usage.

1 To display the heap size (that is, default `datasize`), use one of the following:

The `csh` command:

```
# limit
```

The `ksh` command

```
# ulimit -a
```

The Solaris utility

```
# sysdef
```

2 For each dispatcher service compute `STACKSIZE*MAX_CONNS`, and then add up the values computed for each service. The system's heap size needs to be at least twice this number.

25.4 Managing Message Store, Admin, and Default Service Logs

This section describes logging for the Message Store (POP, IMAP, and HTTP), Admin, and Default services. (see [Table 25–1](#)).

For these services, you specify log settings and to view logs. The settings you specify affect which and how many events are logged. You can use those settings and other characteristics to refine searches for logged events when you are analyzing log files.

This section contains the following subsections:

- “[25.4.1 Understanding Service Log Characteristics](#)” on page 821
- “[25.4.2 Understanding Service Log File Format](#)” on page 823

- “[25.4.3 Defining and Setting Service Logging Options](#)” on page 824
- “[25.4.4 Searching and Viewing Service Logs](#)” on page 826
- “[25.4.5 Working With Service Logs](#)” on page 827
- “[25.4.6 Using Message Tracing for Message Store Logging](#)” on page 830
- “[25.4.7 Other Message Store Logging Features](#)” on page 831
- “[25.4.8 Message Store Logging Examples](#)” on page 831

25.4.1

Understanding Service Log Characteristics

This section describes the following log characteristics for the message store and administration services: logging levels, categories of logged events, filename conventions for logs, and log-file directories.

25.4.1.1

Logging Levels

The level, or priority, of logging defines how detailed, or verbose, the logging activity is to be. A higher priority level means less detail; it means that only events of high priority (high severity) are logged. A lower level means greater detail; it means that more events are recorded in the log file.

You can set the logging level separately for each service—POP, IMAP, HTTP, Admin, and Default by setting the `logfile.service.level` configuration parameter (see “[25.4.3 Defining and Setting Service Logging Options](#)” on page 824). You can also use logging levels to filter searches for log events. Table 25–6 describes the available levels. These logging levels are a subset of those defined by the UNIX `syslog` facility.

TABLE 25–6 Logging Levels for Store and Administration Services

Level	Description
Critical	The minimum logging detail. An event is written to the log whenever a severe problem or critical condition occurs—such as when the server cannot access a mailbox or a library needed for it to run.
Error	An event is written to the log whenever an error condition occurs—such as when a connection attempt to a client or another server fails.
Warning	An event is written to the log whenever a warning condition occurs—such as when the server cannot understand a communication sent to it by a client.
Notice	An event is written to the log whenever a notice (a normal but significant condition) occurs—such as when a user login fails or when a session closes. This is the default log level.
Information	An event is written to the log with every significant action that takes place—such as when a user successfully logs on or off or creates or renames a mailbox.

TABLE 25–6 Logging Levels for Store and Administration Services *(Continued)*

Level	Description
Debug	The most verbose logging. Useful only for debugging purposes. Events are written to the log at individual steps within each process or task, to pinpoint problems.

When you select a particular logging level, events corresponding to that level and to all higher (less verbose) levels are logged. The default level of logging is Notice.

Note – The more verbose the logging you specify, the more disk space your log files will occupy; for guidelines, see “[25.4.3 Defining and Setting Service Logging Options](#)” on page [824](#)

25.4.1.2

Categories of Logged Events

Within each supported service or protocol, Messaging Server further categorizes logged events by the facility, or functional area, in which they occur. Every logged event contains the name of the facility that generated it. These categories aid in filtering events during searches. [Table 25–7](#) lists the categories that Messaging Server recognizes for logging purposes.

TABLE 25–7 Categories in Which Log Events Occur

Facility	Description
General	Undifferentiated actions related to this protocol or service
LDAP	Actions related to Messaging Server accessing the LDAP directory database
Network	Actions related to network connections (socket errors fall into this category)
Account	Actions related to user accounts (user logins fall into this category)
Protocol	Protocol-level actions related to protocol-specific commands (errors returned by POP, IMAP, or HTTP functions fall into this category)
Stats	Actions related to the gathering of server statistics
Store	Low-level actions related to accessing the message store (read/write errors fall into this category)

For examples of using categories as filters in log searches, see “[25.4.4 Searching and Viewing Service Logs](#)” on page [826](#)

25.4.1.3

Service Log File Directories

Every logged service is assigned a single directory, in which its log files are stored. All IMAP log files are stored together, as are all POP log files, and log files of any other service. You define the location of each directory, and you also define how many log files of what maximum size are permitted to exist in the directory.

Make sure that your storage capacity is sufficient for all your log files. Log data can be voluminous, especially at lower (more verbose) logging levels.

It is important also to define your logging level, log rotation, log expiration, and server-backup policies appropriately so that all of your log-file directories are backed up and none of them become overloaded; otherwise, you may lose information. See “[25.4.3 Defining and Setting Service Logging Options](#)” on page 824.

25.4.2 Understanding Service Log File Format

All message store and administration service log files created by Messaging Server have identical content formats. Log files are multiline text files, in which each line describes one logged event. All event descriptions, for each of the supported services, have the general format:

dateTime hostName processName[pid]: category logLevel: eventMessage

Table 25–8 lists the log file components. Note that this format of event descriptions is identical to that defined by the UNIX syslog facility, except that the date/time format is different and the format includes two additional components (*category* and *logLevel*).

TABLE 25–8 Store and Administration Log File Components

Component	Definition
<i>dateTime</i>	The date and time at which the event was logged, expressed in <i>dd/mm/yyyy hh:mm:ss</i> format, with a time-zone field expressed as <i>+/-hhmm</i> from GMT. For example: <code>02/Jan/1999:13:08:21 -0700</code>
<i>hostName</i>	The name of the host machine on which the server is running: for example, <code>showshoe</code> . Note: If there is more than one instance of Messaging Server on the host, you can use the process ID (pid) to separate logged events of one instance from another.
<i>processName</i>	The name of the process that generated the event: for example, <code>cgi_store</code> .
<i>pid</i>	The process ID of the process that generated the event: for example, <code>18753</code> .
<i>category</i>	The category that the event belongs to: for example, <code>General</code> (see Example 25–5).
<i>logLevel</i>	The level of logging that the event represents: for example, <code>Notice</code> (see Example 25–4).
<i>eventMessage</i>	An event-specific explanatory message that may be of any length: for example, <code>Log created (894305624)</code> .

Here are three examples of logged events:

```
02/May/1998:17:37:32 -0700 showshoe cgi_store[18753]:
General Notice:
Log created (894155852)
```

```
04/May/1998:11:07:44 -0400 xyzmail cgi_service[343]: General Error:  
function=getserverhello|port=2500|error=failed to connect
```

```
03/Dec/1998:06:54:32 +0200 SiroePost imapd[232]: Account Notice:  
close [127.0.0.1] [unauthenticated] 1998/12/3 6:54:32  
0:00:00 0 115 0
```

IMAP and POP event entries may end with three numbers. The example above has 0 115 0. The first number is bytes sent by client, the second number is the bytes sent by the server, and third number is mailboxes selected (always 1 for POP).

When viewing a log file in the Log Viewer window, you can limit the events displayed by searching for any specific component in an event, such as a specific logging level or category, or a specific process ID. For more information, see “[25.4.4 Searching and Viewing Service Logs](#)” on page 826.

The event message of each log entry is in a format specific to the type of event being logged, that is, each service defines what content appears in any of its event messages. Many event messages are simple and self-evident; others are more complex.

25.4.3

Defining and Setting Service Logging Options

You can define the message store and administration service logging configurations that best serve your administration needs. This section discusses issues that may help you decide on the best configurations and policies, and it explains how to implement them.

25.4.3.1

Flexible Logging Architecture

The naming scheme for log files (*service.sequenceNum.timeStamp*) helps you to design a flexible log-rotation and backup policy. The fact that events for different services are written to different files makes it easier for you to isolate problems quickly. Also, because the sequence number in a filename is ever-increasing and the timestamp is always unique, later log files do not simply overwrite earlier ones after a limited set of sequence numbers is exhausted. Instead, older log files are overwritten or deleted only when the more flexible limits of age, number of files, or total storage are reached.

Messaging Server supports automatic rotation of log files, which simplifies administration and facilitates backups. You are not required to manually retire the current log file and create a new one to hold subsequent logged events. You can back up all but the current log file in a directory at any time, without stopping the server or manually notifying the server to start a new log file.

In setting up your logging policies, you can set options (for each service) that control limits on total log storage, maximum number of log files, individual file size, maximum file age, and rate of log-file rotation.

25.4.3.2

Planning the Options You Want

Keep in mind that you must set several limits, more than one of which might cause the rotation or deletion of a log file. Whichever limit is reached first is the controlling one. For example, if your maximum log-file size is 3.5 MB, and you specify that a new log be created every day, you may actually get log files created faster than one per day if log data builds up faster than 3.5 MB every 24 hours. Then, if your maximum number of log files is 10 and your maximum age is 8 days, you may never reach the age limit on log files because the faster log rotation may mean that 10 files will have been created in less than 8 days.

The following default values, provided for Messaging Server administration logs, may be a reasonable starting point for planning:

Maximum number of log files in a directory: 10

Maximum log-file size: 2 MB

Total maximum size permitted for all log files: 20 MB

Minimum free disk space permitted: 5 MB

Log rollover time: 1 day

Maximum age before expiration: 7 days

Level of logging: Notice

You can see that this configuration assumes that server-administration log data is predicted to accumulate at about 2 MB per day, backups are weekly, and the total space allotted for storage of admin logs is at least 25 MB. (These settings may be insufficient if the logging level is more verbose.)

For POP, IMAP or HTTP logs, the same values might be a reasonable start. If all services have approximately the same log-storage requirements as the defaults shown here, you might expect to initially plan for about 150 MB of total log-storage capacity. (Note that this is meant only as a general indication of storage requirements; your actual requirements may be significantly different.)

25.4.3.3

Understanding Logging Options

You can set options that control the message store logging configuration by the command line.

The optimal settings for these options depend on the rate at which log data accumulates. It may take between 4,000 and 10,000 log entries to occupy 1 MB of storage. At the more verbose levels of logging (such as Notice), a moderately busy server may generate hundreds of megabytes of log data per week. Here is one approach you can follow:

- Set a level of logging that is consistent with your storage limits—that is, a level that you estimate will cause log-data accumulation at approximately the rate you used to estimate the storage limit.

- Define the log file size so that searching performance is not impacted. Also, coordinate it with your rotation schedule and your total storage limit. Given the rate at which log entries accumulate, you might set a maximum that is slightly larger than what you expect to accumulate by the time a rotation automatically occurs. And your maximum file size times your maximum number of files might be roughly equivalent to your total storage limit.
For example, if your IMAP log rotation is daily, your expected accumulation of IMAP log data is 3 MB per day, and your total storage limit for IMAP logs is 25 MB, you might set a maximum IMAP log-file size of 3.5 MB. (In this example, you could still lose some log data if it accumulated so rapidly that all log files hit maximum size and the maximum number of log files were reached.)
- If server backups are weekly and you rotate IMAP log files daily, you might specify a maximum number of IMAP log files of about 10 (to account for faster rotation if the individual log-size limit is exceeded), and a maximum age of 7 or 8 days.
- Pick a total storage limit that is within your hardware capacity and that coordinates with the backup schedule you have planned for the server. Estimate the rate at which you anticipate that log data will accumulate, add a factor of safety, and define your total storage limit so that it is not exceeded over the period between server backups.
For example, if you expect to accumulate an average of 3 MB of IMAP log-file data per day, and server backups are weekly, you might specify on the order of 25 - 30 MB as the storage limit for IMAP logs (assuming that your disk storage capacity is sufficient).
- For safety, pick a minimum amount of free disk space that you will permit on the volume that holds the log files. That way, if factors other than log-file size cause the volume to fill up, old log files will be deleted before a failure occurs from attempting to write log data to a full disk.

25.4.4

Searching and Viewing Service Logs

The log files provide the basic interface for viewing message store and administration log data. For a given service, log files are listed in chronological order. Once you have chosen a log file to search, you can narrow the search for individual events by specifying search parameters.

25.4.4.1

Search Parameters

These are useful search parameters you can specify for viewing log data:

- **A time period.** You can specify the beginning and end of a specific time period to retrieve events from, or you can specify a number of days (before the present) to search. You might typically specify a range to look at logged events leading up to a server crash or other occurrence whose time you know of. Alternatively, you might specify a day range to look at only today's events in the current log file.
- **A level of logging.** You can specify the logging level (see “[25.4.1.1 Logging Levels](#)” on [page 821](#) example, Critical to see why the server went down, or Error to locate failed protocol calls.

- **A facility.** You can specify the facility (see “[25.4.1.2 Categories of Logged Events](#)” on page 822 that contains the problem; for example, Store if you believe a server crash involved a disk error, or Protocol if the problem lies in an IMAP protocol command error.
- **A text search pattern.** You can provide a text search pattern to further narrow the search. You can include any component of the event (see “[25.4.2 Understanding Service Log File Format](#)” on page 823) search, such as event time, process name, process ID, and any part of the event message (such as remote host name, function name, error number, and so on) that you know defines the event or events you want to retrieve.

Your search pattern can include the following special and wildcard characters:

* Any set of characters (example: *.com)

? Any single character (example: 199?)

[nnn] Any character in the set nnn (example: [aeiou])

[^nnn] Any character not in the set nnn (example: [^aeiou])

[n-m] Any character in the range n-m (example: [A-Z])

[^n-m] Any character not in the range n-m (example: [^0-9])

\ Escape character: place before *, ?, [, or] to use them as literals

Note: Searches are case-sensitive.

Examples of combining logging level and facility in viewing logs might include the following:

- Specifying Account facility (and Notice level) to display failed logins, which may be useful when investigating potential security breaches
- Specifying Network facility (and all logging levels) to investigate connection problems
- Specifying all facilities (and Critical logging level) to look for basic problems in the functioning of the server

25.4.5 Working With Service Logs

This section describes how to work with service logs by using the configutil command for searching and viewing logs. It consists of the following sections:

- “[To Send Service Logs to syslog](#)” on page 828
- “[25.4.5.1 To Disable HTTP Logging](#)” on page 828
- “[To Set the Server Log Level](#)” on page 828
- “[To Specify a Directory Path for Server Log Files](#)” on page 828
- “[To Specify a Maximum File Size for Each Service Log](#)” on page 829
- “[To Specify a Service Log Rotation Schedule](#)” on page 829

- “To Specify a Maximum Number of Service Log Files Per Directory” on page 829
- “To Specify a Storage Limit” on page 829
- “To Specify the Minimum Amount of Free Disk Space to Reserve” on page 829
- “25.4.5.2 To Specify an Age for Logs at Which They Expire” on page 829

▼ To Send Service Logs to syslog

- Run the configutil command with the `syslogfacility` option:

```
configutil -o logfile.service.syslogfacility -v value
```

where *service* is admin, pop, imap, imta, or http and *value* is user, mail, daemon, local0 to local7, or none.

Once the value is set, messages are logged to the syslog facility corresponding to the set value and all the other log file service options are ignored. When the option is not set or the value is none, logging uses the Messaging Server log files.

25.4.5.1

To Disable HTTP Logging

If your system does not support HTTP message access, that is, Webmail, you can disable HTTP logging by setting the following variables. Do not set these variables if your system requires Webmail support (for example, Messenger Express).

- Run the following configutil commands:

```
configutil -o service.http.enable -v no  
configutil -o service.http.enablesslport -v no
```

▼ To Set the Server Log Level

- Run the following configutil command:

```
configutil -o logfile.service.loglevel -v level
```

where *service* is admin, pop, imap, imta, or http and *loglevel* is Nolog, Critical, Error, Warning, Notice, Information, or Debug.

▼ To Specify a Directory Path for Server Log Files

- Run the following configutil command:

```
configutil -o logfile.service.logdir -v filepath
```

▼ To Specify a Maximum File Size for Each Service Log

- Run the following configutil command:

```
configutil -o logfile.service.maxlogfilesize -v size
```

where *size* specifies a number of bytes.

▼ To Specify a Service Log Rotation Schedule

- Run the following configutil command:

```
configutil -o logfile.service.rollovertime -v number
```

where *number* specifies a number of seconds.

▼ To Specify a Maximum Number of Service Log Files Per Directory

- Run the following configutil command:

```
configutil -o logfile.service.maxlogfiles -v number
```

where *number* specifies the maximum number of log files.

▼ To Specify a Storage Limit

- Run the following configutil command:

```
configutil -o logfile.service.maxlogsize -v number
```

where *number* specifies a number in bytes.

▼ To Specify the Minimum Amount of Free Disk Space to Reserve

- Run the following configutil command:

```
configutil -o logfile.service.minfreediskspace -v number
```

where *number* specifies a number in bytes.

25.4.5.2 To Specify an Age for Logs at Which They Expire

```
configutil -o logfile.service.expirytime -v number
```

where *number* specifies a number in seconds.

25.4.6

Using Message Tracing for Message Store Logging

You can use Message Store logging to trace messages by message ID in a way similar to how the MTA traces messages. Tracing messages in this fashion enables you to track the critical events of a message's life cycle.

To trace messages in the Message Store logs, you need to configure message tracing in addition to the normal logging configuration. By default, message tracing is not enabled.

Note – Message tracing will fill up a large amount of disk space. Do not enable this feature unless you have adequate disk space.

Message Store logging can track the following operations:

- append - The primary way the Message Store library adds a message to a folder. Tracing append shows messages entering the Message Store.
- fetch - The IMAP command that retrieves a message or part of a message for the end user. For message tracing, its meaning is extended to when any service retrieves a message for the end user to read.
In message tracing, you might sometimes like to avoid tracking when the header of a message is read, so a body fetch is in reference to when some part of the body of a message is retrieved.
- Expunge: An IMAP term that is extended in this case to reference when any service removes a message from a user's folder.

▼ To Enable Message Tracing

- Run either the following configutil commands:

```
configutil -o local.msgtrace.active -v "yes"
```

Message trace information is written to the default log for each process. The IMAP fetches appear in the `imap` log file. The `ims_master` appends appear in the `ims_master` channel log file.

```
configutil -o local.msgtrace.active -v "msgtrace"
```

In this command message trace information for all processes is written to the `msgtrace` log file

▼ To Configure LMTP Logging

- If you are using LMTP, and not using a single “msgtrace” log file, then you must also locally configure the `tcp_lmtp_server` log file. If you are not using LMTP, or are not using message

trace, or are using message trace in the “msgtrace” log file, you do not need to initialize the LMTP Message Store side log. (LMTP already logs MTA information separately.) For example:

```
configutil -o "local.logfile.tcp_lmtp_server.buffersize" -v "0"
configutil -o "local.logfile.tcp_lmtp_server.expirytime" -v "604800"
configutil -o "local.logfile.tcp_lmtp_server.flushinterval" -v "60"
configutil -o "local.logfile.tcp_lmtp_server.logdir" -v \
    "/opt/SUNWmsgsr/data/log"
configutil -o "local.logfile.tcp_lmtp_server.loglevel" -v "Information"
configutil -o "local.logfile.tcp_lmtp_server.logtype" -v "NscpLog"
configutil -o "local.logfile.tcp_lmtp_server.maxlogfiles" -v "10"
configutil -o "local.logfile.tcp_lmtp_server.maxlogfilesize" -v "2097152"
configutil -o "local.logfile.tcp_lmtp_server.maxlogsize" -v "20971520"
configutil -o "local.logfile.tcp_lmtp_server.minfreediskspace" \
    -v "5242880"
configutil -o "local.logfile.tcp_lmtp_server.rollovertime" -v "86400"
```

25.4.7 Other Message Store Logging Features

Messaging Server provides a feature called telemetry that can capture a user’s entire IMAP or POP session into a file. This feature is useful for debugging client problems. For example, if a user complains that their message access client is not working as expected, this feature can be used to trace the interaction between the access client and Messaging Server. See “[20.14.1.3 Check User IMAP/POP/Webmail Session by Using Telemetry](#)” on page 661

25.4.8 Message Store Logging Examples

The exact field format and list of fields logged in the Message Store log files vary according to the logging options set. This section shows a few examples of interpreting typical sorts of log entries.

- “[25.4.8.1 Message Store Logging Example: Bad Password](#)” on page 831
- “[25.4.8.2 Message Store Logging – Account Disabled](#)” on page 832
- “[25.4.8.3 Message Store Logging Example: Message Appended](#)” on page 832
- “[25.4.8.4 Message Store Logging Example: Message Retrieved by a Client](#)” on page 832
- “[25.4.8.5 Message Store Logging Example: Message Removed from a Folder](#)” on page 832
- “[25.4.8.6 Message Store Logging Example: Duplicate Login Messages](#)” on page 833

25.4.8.1 Message Store Logging Example: Bad Password

When a user types an invalid password, “authentication” failure is logged, as opposed to a “user not found” message. The message “user not found” is the text passed to the client for security reasons, but the real reason (invalid password) is logged.

EXAMPLE 25-11 Message Store Logging – Invalid Password

```
[30/Aug/2004:16:53:05 -0700] vadard imapd[13027]: Account Notice: badlogin:  
[192.18.126.64:40718] plaintext user1 authentication failure
```

25.4.8.2**Message Store Logging – Account Disabled**

The following example shows why a user cannot log in due to a disabled account. Furthermore, the disabled account is clarified as “(inactive)” or “(hold).”

EXAMPLE 25-12 Message Store Logging – Account Disabled

```
[30/Aug/2004:16:53:31 -0700] vadard imapd[13027]: Account Notice: badlogin:  
[192.18.126.64:40720] plaintext user3 account disabled (hold)
```

25.4.8.3**Message Store Logging Example: Message Appended**

The following example shows an append message, which occurs when whenever a message is appended to a folder. The Message Store log records all messages entering the Message Store through the ims_master and lmtp channels. Records the “append” of user ID, folder, message size, and message ID.

EXAMPLE 25-13 Message Store Logging – Append

```
[31/Aug/2004:16:33:14 -0700] vadard ims_master[13822]: Store Information:append:  
user1:user/user1:659:<Roam.SIMC.2.0.6.1093995286.11265.user1@vadar.siroe.com>
```

25.4.8.4**Message Store Logging Example: Message Retrieved by a Client**

The Message Store log writes a “fetch” message when a client retrieves a message. The Message Store log records all client fetches of at least one body part. Records the “fetch” of user ID, folder, and message-ID.

EXAMPLE 25-14 Message Store Logging – Message Retrieved by a Client

```
[31/Aug/2004:15:55:26 -0700] vadard imapd[13729]: Store Information:  
fetch:user1:user/user1:<Roam.SIMC.2.0.6.1093051161.3655.user1@vad.siroe.com>
```

25.4.8.5**Message Store Logging Example: Message Removed from a Folder****EXAMPLE 25-15** Message Store Logging Example: Message Removed from a Folder

The Message Store writes an “expunge” message when an IMAP or POP message is removed from a folder (but not removed from the system). It is logged whether it is expunged by the user or a utility. Records the “expunge” of folder and message ID.

```
31/Aug/2004:16:57:36 -0700] vadard imexpire[13923]: Store Information:  
expunge:user/user1:<Roam.SIMC.2.0.6.1090458838.2929.user1@vadar.siroe.com>
```

25.4.8.6 Message Store Logging Example: Duplicate Login Messages

If you configure message trace for one msgtrace log file, the normal “login” messages, which appear in the imap and pop log files, are duplicated in the msgtrace file. The following is a normal login message:

EXAMPLE 25-16 Message Store Logging – Login

```
[30/Aug/2004:16:53:13 -0700] vadar imapd[13027]: Account Information: login  
[192.18.126.64:40718] user1 plaintext
```


Troubleshooting the MTA

This chapter describes common tools, methods, and procedures for troubleshooting the Message Transfer Agent (MTA). It consists of the following sections:

- “[26.1 Troubleshooting Overview](#)” on page 835
- “[26.2 Standard MTA Troubleshooting Procedures](#)” on page 836
- “[26.3 Common MTA Problems and Solutions](#)” on page 846
- “[26.4 General Error Messages](#)” on page 859

A related topic, monitoring procedures can be found in [Chapter 27, “Monitoring Messaging Server”](#)

Note – Prior to reading this chapter, you should review Chapters 5 through 10 in this guide and the MTA configuration and command-line utility chapters in the *Sun Java System Messaging Server Administration Reference*.

26.1 Troubleshooting Overview

One of the first steps in troubleshooting the MTA is to determine where to begin the diagnosis. Depending on the problem, you might look for error messages in log files. In other situations, you might check all of the standard MTA processes, review the MTA configuration, or start and stop individual channels. Whatever approach you use, consider the following questions when troubleshooting the MTA:

- Did configuration or environmental problems prevent messages from being accepted (for example, disk space or quota problems)?
- Were MTA services such as the Dispatcher and the Job Controller present at the time the message entered the message queue?
- Did network connectivity or routing problems cause messages to be stuck or misrouted on a remote system?

- Did the problem occur before or after a message entered into the message queue?

This chapter will address these questions in the subsequent sections.

26.2 Standard MTA Troubleshooting Procedures

This section outlines standard troubleshooting procedures for the MTA. Follow these procedures if a problem does not generate an error message, if an error message does not provide enough diagnostic information, or if you want to perform general wellness checks, testing, and standard maintenance of the MTA.

- “[26.2.1 Check the MTA Configuration](#)” on page 836
- “[26.2.2 Check the Message Queue Directories](#)” on page 836
- “[26.2.3 Check the Ownership of Critical Files](#)” on page 837
- “[26.2.4 Check that the Job Controller and Dispatcher are Running](#)” on page 837
- “[26.2.5 Check the Log Files](#)” on page 838
- “[26.2.6 Run a Channel Program Manually](#)” on page 839
- “[26.2.7 Starting and Stopping Individual Channels](#)” on page 840
- “[26.2.8 An MTA Troubleshooting Example](#)” on page 842

26.2.1 Check the MTA Configuration

Test your address configuration by using the `imsimta test -rewrite` utility. With this utility, you can test the MTA’s address rewriting and channel mapping without actually having to send a message. Refer to the MTA Command-line Utilities chapter in the [Chapter 2, “Message Transfer Agent Command-line Utilities,” in Sun Java System Messaging Server 6.3 Administration Reference](#) for more information.

The utility will normally show address rewriting that will be applied as well as the channel to which messages will be queued. However, syntax errors in the MTA configuration will cause the utility to issue an error message. If the output is not what you expect, you may need to correct your configuration.

26.2.2 Check the Message Queue Directories

Check if messages are present in the MTA message queue directory, typically `msg-svr-base/data/queue/`. Use command-line utilities like `imsimta qm` to check for the presence of expected message files under the MTA message queue directory. For more information on `imsimta qm`, refer to the MTA command-line utilities chapter in the [“imsimta qm” in Sun Java System Messaging Server 6.3 Administration Reference](#) and [“27.8.6 imsimta qm counters” on page 885](#)

If the `imsimta test -rewrite` output looks correct, check that messages are actually being placed in the MTA message queue subdirectories. To do so, enable message logging (For more information on MTA logging, see “[25.3 Managing MTA Message and Connection Logs](#)” on page 796 in the directory `/msg-svr-base/log/`). You can track a specific message by its message ID to ensure that it is being placed in the MTA message queue subdirectories. If you are unable to find the message, you may have a problem with file disk space or directory permissions.

26.2.3 Check the Ownership of Critical Files

You should have selected a mail server user account (`mailsrv` by default) when you installed Messaging Server. The following directories, subdirectories, and files should be owned by this account:

```
msg-svr-base/data/queue/
msg-svr-base/data/log
msg-svr-base/data/tmp
```

Commands, like the ones in the following UNIX system example, may be used to check the protection and ownership of these directories:

```
ls -l -p -d /opt/SUNWmsgsr/data/queue
drwxr-x--- 2 mailsrv mail 512 Jan  4 16:09 /opt/SUNWmsgsr/data/queue/

ls -l -p -d /opt/SUNWmsgsr/data/log
drwxr-x--- 2 mailsrv mail 3072 Feb 16 12:07 /opt/SUNWmsgsr/data/log/

ls -l -p -d /opt/SUNWmsgsr/data/tmp
drwxr-x--- 2 mailsrv mail 512 Feb 16 12:55 /opt/SUNWmsgsr/data/tmp/
```

Check that the files in `msg-svr-base/data/queue` are owned by the MTA account by using commands like in the following UNIX system example:

```
ls -l -p -R /opt/SUNWmsgsr/data/queue
```

26.2.4 Check that the Job Controller and Dispatcher are Running

The MTA Job Controller handles the execution of the MTA processing jobs, including most outgoing (master) channel jobs.

Some MTA channels, such as the MTA’s multi-threaded SMTP channels, include resident server processes that process incoming messages. These servers handle the slave (incoming) direction for the channel. The MTA Dispatcher handles the creation of such MTA servers. Dispatcher configuration options control the availability of the servers, the number of created servers, and how many connections each server can handle.

To check that the Job Controller and Dispatcher are present, and to see if there are MTA servers and processing jobs running, use the command `imsimta process`. Under idle conditions the command should result in `job_controller` and `dispatcher` processes. For example:

```
# imsimta process
USER      PID S VSZ   RSS   STIME    TIME     COMMAND
mailsrv  9567 S 18416 9368  02:00:02 0:00   /opt/SUNWmsgsr/lib/tcp_smtp_server
mailsrv  6573 S 18112 5720  Jul_13   0:00   /opt/SUNWmsgsr/lib/job_controller
mailsrv  9568 S 18416 9432  02:00:02 0:00   /opt/SUNWmsgsr/lib/tcp_smtp_server
mailsrv  6574 S 17848 5328  Jul_13   0:00   /opt/SUNWmsgsr/lib/dispatcher
```

If the Job Controller is not present, the files in the `/msg-svr-base/data/queue` directory will get backed up and messages will not be delivered. If you do not have a Dispatcher, then you will be unable to receive any SMTP connections.

For more information on `imsimta process`, refer to the “[imsimta process](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

You could also use `imsimta qm` jobs to list, channel by channel, all active and pending delivery processing jobs currently being managed by the Job Controller. Additional cumulative information is provided for each channel such as the number of message files successfully delivered and those requeued for subsequent delivery attempts. The command syntax is as follows:

```
jobs [-[no]hosts] [-[no]jobs] [-[no]messages] [channel-name]
```

If neither the Job Controller nor the Dispatcher is present, you should review the `dispatcher.log-*` or `job_controller.log-*` file in `/msg-svr-base/data/log`

If the log files do not exist or do not indicate an error, start the processes by using the `start-msg` command. For more information, refer to the MTA command-line utilities chapter in the “[start-msg](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

Note – You should not see multiple instances of the Dispatcher or Job Controller when you run `imsimta process`, unless the system is in the process of forking (`fork()`) child processes before it executes (`exec()`) the program that needs to run. However, the time frame during such duplication is very small.

26.2.5 Check the Log Files

If MTA processing jobs run properly but messages stay in the message queue directory, you can examine the log files to see what is happening. All MTA log files are created in the directory `/msg-svr-base/log`. Log file name formats for various MTA processing jobs are shown in [Table 26–1](#).

TABLE 26-1 MTA Log Files

File Name	Log File Contents
<i>channel_master.log-uniqueid</i>	Output of master program (usually client) for <i>channel</i> .
<i>channel_slave.log-uniqueid</i>	Output of slave program (usually server) for <i>channel</i> .
<i>dispatcher.log-uniqueid</i>	Dispatcher debugging. This log is created regardless if the Dispatcher DEBUG option is set. However, to get detailed debugging information, you should set the DEBUG option to a non-zero value.
<i>imta</i>	<i>ims-ms</i> channel error messages when there is a problem in delivery.
<i>job_controller.log-uniqueid</i>	Job controller logging. This log is created regardless if the Job Controller DEBUG option is set. However, to get detailed debugging information, you should set the DEBUG option to a non-zero value.
<i>tcp_smtp_server.log-uniqueid</i>	Debugging for the <i>tcp_smtp_server</i> . The information in this log is specific to the server, not to messages.
<i>return.log-uniqueid</i>	Debug output for the periodic MTA message bouncer job; this log file is created if the <i>return_debug</i> option is used in the <i>option.dat</i>

Note – Each log file is created with a unique ID (*uniqueid*) to avoid overwriting an earlier log created by the same channel. To find a specific log file, you can use the *imsimta* view utility. You can also purge older log files by using the *imsimta* purge command. Note, however, that by default this command is run on a regular basis (see “[4.6.2 Pre-defined Automatic Tasks](#)” on page 130). For more information, see the MTA command-line utilities chapter in the “[imsimta purge](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

The *channel_master.log-uniqueid* and *channel_slave.log-uniqueid* log files will be created in any of the following situations:

- There are errors in your current configuration.
- The *master_debug* or *slave_debug* keywords are set on the channel in the *imta.cnf* file.
- If *mm_debug* is set to a non-zero value (*mm_debug > 0*) in your *option.dat* file (in the directory: */msg-svr-base/config/*).

For more information on debugging channel master and slave programs, see the *Sun Java System Messaging Server Administration Reference*.

26.2.6 Run a Channel Program Manually

When diagnosing an MTA delivery problem it is helpful to manually run an MTA delivery job, particularly after you enable debugging for one or more channels.

The command `imsimta submit` will notify the MTA Job Controller to run the channel. If debugging is enabled for the channel in question, `imsimta submit` will create a log file in directory `/msg-svr-base/log` as shown in [Table 26–1](#).

The command `imsimta run` will perform outbound delivery for the channel under the currently active process, with output directed to your terminal. This may be more convenient than submitting a job, particularly if you suspect problems with job submission itself.

Note – In order to manually run channels, the Job Controller must be running.

For information on syntax, options, parameters, examples of `imsimta submit` and `imsimta run` commands, refer to “[Command Descriptions](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

26.2.7 Starting and Stopping Individual Channels

In some cases, stopping and starting individual channels may make message queue problems easier to diagnose and debug. Stopping a message queue allows you to examine queued messages to determine the existence of loops or spam attacks.

▼ To Stop Outbound Processing (dequeuing) for a Specific Channel

- 1 Use the `imsimta qm stop` command to stop a specific channel. Doing so prevents you from having to stop the Job Controller and having to recompile the configuration. In the following example, the conversion channel is stopped:

`imsimta qm stop conversion`

- 2 To resume processing, use the `imsimta qm start` command to restart the channel. In the following example, the conversion channel is started:

`imsimta qm start conversion`

For more information on the `imsimta qm start` and `imsimta qm stop` commands, see “[imsimta qm](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

Note – The command `imsimta qm start/stop channel` may fail if run simultaneously for many channels at the same time. The tool might have trouble updating the `hold_list` and could report: `QM-E-NOTSTOPPED`, unable to stop the channel; cannot update the hold list. `imsimta qm start/stop channel` should only be used sequentially with a few seconds interval between each run.

If you only want the channel to run between certain hours, use the following options in the channel definition section in the job controller configuration file:

```
urgent_delivery=08:00-20:00
normal_delivery=08:00-20:00
nonurgent_delivery=08:00-20:00
```

26.2.7.1 To Stop Inbound Processing from a Specific Domain or IP Address (enqueueing to a channel)

You can run one of the following processes if you want to stop inbound message processing for a specific domain or IP address, while returning temporary SMTP errors to client hosts. By doing so, messages will not be held on your system. Refer to the “[18.1 PART 1. MAPPING TABLES](#)” on page 541.

- To stop inbound processing for a specific host or domain name, add the following access rule to the `ORIG_SEND_ACCESS` mapping table in the MTA mappings file (typically `/msg-svr-base/config/mappings`):

ORIG_SEND_ACCESS

```
*|*@sesta.com|*|* $X4.2.1|$NHost$ temporarily$ blocked
```

By using this process, the sender’s remote MTA will hold messages on their systems, continuing to resend them periodically until you restart inbound processing.

- To stop inbound processing for a specific IP address, add the following access rule to the `PORT_ACCESS` mapping table in the MTA mappings file (typically `/msg-svr-base/config/mappings`):

PORT_ACCESS

```
TCP|*|25|IP_address_to_block|* $N500$ can't$ connect$ now
```

When you want to restart inbound processing from the domain or IP address, be sure to remove these rules from the mapping tables and recompile your configuration. In addition, you may want to create unique error messages for each mapping table. Doing so will enable you to determine which mapping table is being used.

26.2.8 An MTA Troubleshooting Example

This section explains how to troubleshoot a particular MTA problem step-by-step. In this example, a mail recipient did not receive an attachment to an email message. Note: In keeping with MIME protocol terminology, the “attachment” is referred to as a “message part” in this section. The aforementioned troubleshooting techniques are used to identify where and why the message part disappeared (See “[26.2 Standard MTA Troubleshooting Procedures](#)” on [page 836](#)). By using the following steps, you can determine the path the message took through the MTA. In addition, you can determine if the message part disappeared before or after the message entered the message queue. To do so, you will need to manually stop and run channels, capturing the relevant files.

Note – The Job Controller must be running when you manually run messages through the channels.

26.2.8.1

Identify the Channels in the Message Path

By identifying which channels are in the message path, you can apply the `master_debug` and `slave_debug` keywords to the appropriate channels. These keywords generate debugging output in the channels’ master and slave log files; in turn, the master and slave debugging information will assist in identifying the point where the message part disappeared.

1. Add `log_message_id=1` in the `option.dat` file in directory `/msg-svr-base/config`. With this parameter, you will see message ID: header lines in the `mail.log_current` file.
2. Run `imsimta cnbuild` to recompile the configuration.
3. Run `imsimta restart dispatcher` to restart the SMTP server.
4. Have the end user resend the message with the message part.
5. Determine the channels that the message passes through.

While there are different approaches to identifying the channels, the following approach is recommended:

- a. On UNIX platforms, use the `grep` command to search for message ID: header lines in the `mail.log_current` file in directory `/msg-svr-base/log`.
- b. Once you find the message ID: header lines, look for the E (enqueue) and D (dequeue) records to determine the path of the message. Refer to “[25.3.1 Understanding the MTA Log Entry Format](#)” on [page 796](#) for more information on logging entry codes. See the following E and D records for this example:

```
29-Aug-2001 10:39:46.44  tcp_local conversion      E 2 ...
29-Aug-2001 10:39:46.44  conversion tcp_intranet   E 2 ...
29-Aug-2001 10:39:46.44  tcp_intranet          D 2 ...
```

The channel on the left is the source channel, and the channel on the right is the destination channel. In this example, the E and D records indicate that the message's path went from the `tcp_local` channel to the conversion channel and finally to the `tcp_intranet` channel.

26.2.8.2

Manually Start and Stop Channels to Gather Data

This section describes how to manually start and stop channels. See “[26.2.7 Starting and Stopping Individual Channels](#)” on page 840 starting and stopping the channels in the message's path, you are able to save the message and log files at different stages in the MTA process. These files are later used to “[To Identify the Point of Message Breakdown](#)” on page 845.

▼ To Manually Start and Stop Channels

- 1 Set the `mm_debug=5` in the `option.dat` file in directory `/msg-svr-base/config` in order to provide substantial debugging information.
- 2 Add the `slave_debug` and `master_debug` keywords to the appropriate channels in the `imta.cnf` file in directory `/msg-svr-base/config`.
 - a. Use the `slave_debug` keyword on the inbound channel (or any channel where the message is switched to during the initial dialog) from the remote system that is sending the message with the message part. In this example, the `slave_debug` keyword is added to the `tcp_local` channel.
 - b. Add the `master_debug` keyword to the other channels that the message passed through and were identified in “[26.2.8.1 Identify the Channels in the Message Path](#)” on page 842 would be added to the conversion and `tcp_intranet` channels.
 - c. Run the command `imsimta restart dispatcher` to restart the SMTP server.
- 3 Use the `imsimta qm stop` and `imsimta qm start` commands to manually start and stop specific channels. For more on information by using these keywords, see “[26.2.7 Starting and Stopping Individual Channels](#)” on page 840.
- 4 To start the process of capturing the message files, have the end user resend the message with the message part.
- 5 When the message enters a channel, the message will stop in the channel if it has been stopped with the `imsimta qm stop` command. For more information, see Step [Step 3](#).
 - a. Copy and rename the message file before you manually run the next channel in the message's path. See the following UNIX platform example:

```
# cp ZZ01K7LXW76T709TD0TB.00 ZZ01K7LXW76T709TD0TB.KEEP1
```

The message file typically resides in directory similar to `/msg-svr-base/data/queue/destination_channel/001`. The *destination_channel* is the next channel that the message passes through (such as: `tcp_intranet`). If you want to create subdirectories (like `001`, `002`, and so on) in the *destination_channel* directory, add the `subdirs` keyword to the channels.

- b. It is recommended that you number the extensions of the message each time you trap and copy the message in order to identify the order in which the message is processed.
- 6 Resume message processing in the channel and enqueue to the next destination channel in the message's path. To do so, use the `imsimta qm start` command.
- 7 Copy and save the corresponding channel log file (for example: `tcp_intranet_master.log-*`) located in directory `/msg-svr-base/log`. Choose the appropriate log file that has the data for the message you are tracking. Make sure that the file you copy matches the timestamp and the subject header for the message as it comes into the channel. In the example of the `tcp_intranet_master.log-*`, you might save the file as `tcp_intranet_master.keep` so the file is not deleted.
- 8 Repeat steps 5 - 7 until the message has reached its final destination.

The log files you copied in Step 7 should correlate to the message files that you copied in Step 5. If, for example, you stopped all of the channels in the missing message part scenario, you would save the `conversion_master.log-*` and the `tcp_intranet_master.log-*` files. You would also save the source channel log file `tcp_local_slave.log-*`. In addition, you would save a copy of the corresponding message file from each destination channel:
`ZZ01K7LXW76T709TD0TB.KEEP1` from the conversion channel and
`ZZ01K7LXW76T709TD0TB.KEEP2` from the `tcp_intranet` channel.
- 9 Remove debugging options once the message and log files have been copied.

 - a. Remove the `slave_debug` and the `master_debug` keywords from the appropriate channels in the `imta.cnf` file in directory `/msg-svr-base/config`.
 - b. Reset the `mm_debug=0`, and remove `log_message_id=1` in the `option.dat` file in directory `/msg-svr-base/config`.
 - c. Recompile the configuration by using `imsimta cnbuild`.
 - d. Run the command `imsimta restart dispatcher` to restart the SMTP server.

▼ To Identify the Point of Message Breakdown

- 1 By the time you have finished starting and stopping the channel programs, you should have the following files with which you can use to troubleshoot the problem:
 - a. All copies of the message file (for example: ZZ01K7LXW76T709TD0TB.KEEP1) from each channel program
 - b. A `tcp_local_slave.log-*` file
 - c. A set of `channel_master.log-*` files for each destination channel
 - d. A set of `mail.log_current` records that show the path of the message

All files should have timestamps and message ID values that match the message ID: header lines in the `mail.log_current` records. Note that the exception is when messages are bounced back to the sender; these bounced messages will have a different message ID value than the original message.

- 2 Examine the `tcp_local_slave.log-*` file to determine if the message had the message part when it entered the message queue.

Look at the SMTP dialog and data to see what was sent from the client machine.

If the message part did not appear in the `tcp_local_slave.log-*` file, then the problem occurred before the message entered the MTA. As a result, the message was enqueued without the message part. If this the case, the problem could have occurred on the sender's remote SMTP server or in the sender's client machine.

- 3 Investigate the copies of the message files to see where the message part was altered or missing.

If any message file showed that the message part was altered or missing, examine the previous channel's log file. For example, you should look at the `conversion_master.log-*` file if the message part in the message entering the `tcp_intranet` channel was altered or missing.

- 4 Look at the final destination of the message.

If the message part looks unaltered in the `tcp_local_slave.log`, the message files (for example: ZZ01K7LXW76T709TD0TB.KEEP1), and the `channel_master.log-*` files, then the MTA did not alter the message and the message part is disappearing at the next step in the path to its final destination.

If the final destination is the `ims-ms` channel (the Message Store), then you might download the message from the server to a client machine to determine if the message part is being dropped during or after this transfer. If the destination channel is a `tcp_*` channel, then you need to go to the MTA in the message's path. Assuming it is an Messaging Server MTA, you will need to repeat the entire troubleshooting process (See "[26.2.8.1 Identify the Channels in the Message](#)

Path” on page 842, “26.2.8.2 Manually Start and Stop Channels to Gather Data” on page 843, and this section). If the other MTA is not under your administration, then the user who reported the problem should contact that particular site.

26.3 Common MTA Problems and Solutions

This sections lists common problems and solutions for MTA configuration and operation.

- “26.3.1 TLS Problems” on page 846
- “26.3.2 Changes to Configuration Files or MTA Databases Do Not Take Effect” on page 847
- “26.3.3 The MTA Sends Outgoing Mail but Does Not Receive Incoming Mail” on page 847
- “26.3.4 Dispatcher (SMTP Server) Won’t Start Up” on page 847
- “26.3.5 Timeouts on Incoming SMTP connections” on page 848
- “26.3.6 Messages are Not Dequeued” on page 849
- “26.3.7 MTA Messages are Not Delivered” on page 852
- “26.3.8 Messages are Looping” on page 853
- “26.3.9 Received Message is Encoded” on page 856
- “26.3.10 Server-Side Rules (SSR) Are Not Working” on page 857
- “26.3.11 Slow Response After Users Press Send Email Button” on page 858
- “26.3.12 Asterisks in the Local Parts of Addresses or Received Fields” on page 859

26.3.1 TLS Problems

If, during SMTP dialog, the STARTTLS command returns the following error:

454 4.7.1 TLS library initialization failure

and if you have certificates installed and working for pop/imap access, check the following:

- Protections/ownerships of the certificates have to be set so `mailsrv` account can access the files
- The directory where the certificates are stored need to have protections/ownerships set such that the `mailsrv` account can access the files within that directory.

After changing protections and installing certificates, you must run:

```
stop-msg dispatcher  
start-msg dispatcher
```

Restarting should work, but it is better to shut it down completely, install the certificates, and then start things back up.

26.3.2

Changes to Configuration Files or MTA Databases Do Not Take Effect

If changes to your configuration, mapping, conversion, security, option, or alias files are not taking effect, check to see if you have performed the following steps:

1. Recompile the configuration (by running `imsimta cnbuild`).
2. Restart the appropriate processes (like `imsimta restart dispatcher`).
3. Re-establish any client connections.

26.3.3

The MTA Sends Outgoing Mail but Does Not Receive Incoming Mail

Most MTA channels depend upon a slave or channel program to receive incoming messages. For some transport protocols that are supported by the MTA (like TCP/IP and UUCP), you need to make sure that the transport protocol activates the MTA slave program rather than its standard server. Replacing the native `sendmail` SMTP server with the MTA SMTP server is performed as a part of the Messaging Server installation.

For the multi-threaded SMTP server, the startup of the SMTP server is controlled by the Dispatcher. If the Dispatcher is configured to use a `MIN_PROCS` value greater than or equal to one for the SMTP service, then there should always be at least one SMTP server process running (and potentially more, according to the `MAX_PROCS` value for the SMTP service). The `imsimta` process command may be used to check for the presence of SMTP server processes. See “[imsimta process](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for more information.

26.3.4

Dispatcher (SMTP Server) Won’t Start Up

If the dispatcher won’t start up, first check the `dispatcher.log-*` for relevant error messages. If the log indicates problems creating or accessing the `/tmp/.SUNWmsgsr.dispatcher.socket` file, then verify that the `/tmp` protections are set to 1777. This would show up in the permissions as follows:

```
drwxrwxrwt 8 root sys 734 Sep 17 12:14 tmp/
```

Also do an `ls -l` of the `.SUNWmsgsr.dispatcher.socket` file and confirm the proper ownership. For example, if this is created by `root`, then it is inaccessible by `inetmail`.

Do not remove the `.SUNWmsgsr.dispatcher` file and do not create it if it’s missing. The dispatcher will create the file. If protections are not set to 1777, the dispatcher will not start or restart because it won’t be able to create/access the socket file. In addition, there may be other problems occurring not related to the Messaging Server.

26.3.5 Timeouts on Incoming SMTP connections

Timeouts on incoming SMTP connections are most often related to system resources and their allocation. The following techniques can be used to identify the causes of timeouts on incoming SMTP connections:

▼ To Identify the Causes of Timeouts on Incoming SMTP Connections

- 1 Check how many simultaneous incoming SMTP connections you allow. This is controlled by the MAX_PROCS and MAX_CONNS Dispatcher settings for the SMTP service; the number of simultaneous connections allowed is MAX_PROCS*MAX_CONNS. If you can afford the system resources, consider raising this number if it is too low for your usage.
- 2 Another technique you can use is to open a TELNET session.

In the following example, the user connects to 127.0.0.1 port 25. Once connected, 220 banner is returned. For example:

```
telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 budgie.sesta.com --Server ESMTP (Sun Java System Messaging Server 6.1
(built May 7 2001))
```

If you are connected and receive a 220 banner, but additional commands (like ehlo and mail from) do not illicit a response, then you should run imsmta test - rewrite to ensure that the configuration is correct.

- 3 If the response time of the 220 banner is slow, and if running the pstack command on the SMTP server shows the following iii_res* functions (these functions indicate that a name resolution lookup is being performed):

```
febe2c04 iii_res_send (fb7f4564, 28, fb7f4de0, 400, fb7f458c, fb7f4564) +
42c febd9dcc iii_res_query (0, fb7f4564, c, fb7f4de0, 400, 7f) + 254
```

then it is likely that the host has to do reverse name resolution lookups, even on a common pair like localhost/127.0.0.1. To prevent such a performance slowdown, you should reorder your host's lookups in the /etc/nsswitch.conf file. To do so, change the following line in the /etc/nsswitch.conf file from:

```
hosts: dns nis [NOTFOUND=return] files
to:
```

```
hosts: files dns nis [NOTFOUND=return]
```

Making this change in the `/etc/nsswitch.conf` file can improve performance as fewer SMTP servers have to handle messages instead of multiple SMTP servers having to perform unnecessary lookups.

- 4 You can also put the `slave_debug` keyword on the channels handling incoming SMTP over TCP/IP mail, usually `tcp_local` and `tcp_intranet`. After doing so, review the most recent `tcp_local_slave.log-uniqueid` files to identify any particular characteristics of the messages that time out. For example, if incoming messages with large numbers of recipients are timing out, consider using the `expand_limit` keyword on the channel.

Remember that if your system is overloaded and overextended, timeouts will be difficult to avoid entirely.

26.3.6 Messages are Not Dequeued

Errors encountered during TCP/IP delivery are often transient; the MTA will generally retain messages when problems are encountered and retry them periodically. It is normal on large networks to experience periodic outages on certain hosts while other host connections work fine. To verify the problem, examine the log files for errors relating to delivery attempts. You may see error messages such as, “Fatal error from `smtp_open`.” Such errors are not uncommon and are usually associated with a transient network problem. To debug TCP/IP network problems, use utilities like PING, TRACEROUTE, and NSLOOKUP.

The following example shows the steps you might use to see why a message is sitting in the queue awaiting delivery to `xtel.co.uk`. To determine why the message is not being dequeued, you can recreate the steps the MTA uses to deliver SMTP mail on TCP/IP.

```
% nslookup -query=mx xtel.co.uk      (Step 1)

Server: LOCALHOST
Address: 127.0.0.1

Non-authoritative answer:
XTEL.CO.UK  preference = 10, mail exchanger = nsfnet-relay.ac.uk      (Step 2)
```

```
% telnet nsfnet-relay.ac.uk 25      (Step 3)
Trying... [128.86.8.6]
telnet: Unable to connect to remote host: Connection refused
```

1. Use the NSLOOKUP utility to see what MX records, if any, exist for this host. If no MX records exist, then you should try connecting directly to the host. If MX records do exist, then you must connect to the designated MX relays. The MTA honors MX information preferentially, unless explicitly configured not to do so. See also “[12.4.3.5 TCP/IP MX Record Support](#)” on page 373.

2. In this example, the DNS (Domain Name Service) returned the name of the designated MX relay for `xtel.co.uk`. This is the host to which the MTA will actually connect. If more than one MX relay is listed, the MTA will try each MX record in succession, with the lowest preference value tried first.
3. If you do have connectivity to the remote host, you should check if it is accepting inbound SMTP connections by using TELNET to the SMTP server port 25.

Note – If you use TELNET without specifying the port, you will discover that the remote host accepts normal TELNET connections. This does not indicate that it accepts SMTP connections; many systems accept regular TELNET connections but refuse SMTP connections and vice versa. Consequently, you should always do your testing against the SMTP port.

In the previous example, the remote host is refusing connections to the SMTP port. This is why the MTA fails to deliver the message. The connection may be refused due to a misconfiguration of the remote host or some sort of resource exhaustion on the remote host. In this case, nothing can be done locally to resolve the problem. Typically, you should let the MTA continue to retry the message.

If you are running Messaging Server on a TCP/IP network that does not use DNS, you can skip the first two steps. Instead, you can use TELNET to directly access the host in question. Be careful to use the same host name that the MTA would use. Look at the relevant log file from the MTA's last attempt to determine the host name. If you are using host files, you should make sure that the host name information is correct. It is strongly recommended that you use DNS instead of host names.

Note that if you test connectivity to a TCP/IP host and encounter no problems using interactive tests, it is quite likely that the problem has simply been resolved since the MTA last tried to deliver the message. You can re-run the `imsimta submit tcp_channel` on the appropriate channel to see if messages are being dequeued.

26.3.6.1

To Create a New Channel

In certain circumstances, a remote domain can break down and the volume of mail addressed to this server can be so great that the outgoing channel queue will fill up with messages that cannot be delivered. The MTA tries to redeliver these messages periodically (the frequency and number of the retries is configurable using the `backoff` keywords) and under normal circumstances, no action is needed. However, if too many messages get stuck in the queue, other messages may not get delivered in a timely manner because all the channel jobs are working to process the backlog of messages that cannot be delivered.

In this situation, you can reroute these messages to a new channel running in its own job controller pool. This will avoid contention for processing and allow the other channels to deliver their messages. This procedure is described below. We assume a domain called `siroe.com`

▼ To Create a New Channel

1 Create a new channel called `tcp_siroe-daemon` and add a new value for the `pool` keyword.

Channels are created in the channel block section of `/msg-svr-base/config/imta.cnf`. The channel should have the same channel keywords on your regular outgoing `tcp_*` channel. Typically, this is the `tcp_local` channel, which handles all outbound (internet) traffic. Since `siroe.com` is out on the internet, this is the channel to emulate. The new channel may look something like this:

```
tcp_siroe smtp nomx single_sys remotehost inner allowswitchchannel      \
dentnonnumeric subdirs 20 maxjobs 7 pool SMTP_SIROE maytlsserver      \
maysaslserver saslswitchchannel tcp_auth missingrecipientpolicy 0      \
tcp_siroe-daemon
```

Note the new keyword-value pair `pool SMTP_SIROE`. This specifies that messages to this channel will only use computer resources from the `SMTP_SIROE` pool. Note also that a blank line is required before and after the new channel.

2 Add two rewrite rules to the rewrite rule section of the `imta.cnf` file to direct email destined for `siroe.com` to the new channel.

The new rewrite rules look like this:

```
siroe.com      $U%$D@tcp_siroe-daemon
.siroe.com    $U%$H$D@tcp_siroe-daemon
```

These rewrite rules will direct messages to `siroe.com` (including addresses like `host1.siroe.com` or `hostA.host1.siroe.com`) to the new channel whose official host name is `tcp_siroe-daemon`. The rewriting part of these rules, `$U%$D` and `$U%$H$D`, retain the original addresses of the messages. `$U` copies the user name from original address. `%` is the separator—the `@` between the username and domain. `$H` copies the unmatched portion of host/domain specification at the left of dot in pattern. `$D` copies the portion of domain specification that matched.

3 Define a new job controller pool called `SMTP_SIROE`.

In `/msg-svr-base/config/job_controller.cnf` add the following:

```
[POOL=SMTP_SIROE]
job_limit=10
```

This creates a message resource pool called `SMTP_SIROE` that allows up to 10 jobs to be simultaneously run. Be sure not to leave any blank lines between this pool definition and the others. See “[8.7 The Job Controller](#)” on page 202 for details on jobs and pools.

4 Restart the MTA.

Issue the commands: `imsimta cnbuild;imsimta restart`

This recompiles the configuration and restarts the job controller and dispatcher.

In this example, a large quantity of email from your internal users is destined for a particular remote site called `siroe.com`. For some reason, `siroe.com`, is temporarily unable to accept incoming SMTP connections and thus cannot deliver email. (This type of situation is not a rare occurrence.)

As email destined for `siroe.com` comes in, the outgoing channel queue, typically `tcp_local`, will fill up with messages that cannot be delivered. The MTA tries to redeliver these messages periodically (the frequency and number of the retries is configurable using the `backoff` keywords) and under normal circumstances, no action is needed.

However, if too many messages get stuck in the queue, other messages may not get delivered in a timely manner because all the channel jobs are working to process the backlog of `siroe.com` messages. In this situation, you may wish reroute `siroe.com` messages to a new channel running in its own job controller pool (see “[8.7 The Job Controller](#)” on page 202). This will allow the other channels to deliver their messages without having to contend for processing resources used by `siroe.com` messages. Creating a new channel to address this situation is described below.

26.3.7 MTA Messages are Not Delivered

In addition to message transport problems, there are two common problems which can result in unprocessed messages in the message queues:

1. The queue cache is not synchronized with the messages in the queue directories. Message files in the MTA queue subdirectories that are awaiting delivery are entered into an in-memory queue cache. When channel programs run, they consult this queue cache to determine which messages to deliver in their queues. There are circumstances where there are message files in the queue, but there is no corresponding queue cache entry.
 - a. To check if a particular file is in the queue cache, you can use the `imsimta cache -view` utility; if the file is not in the queue cache, then the queue cache needs to be synchronized.

The queue cache is normally synchronized every four hours. If required, you can manually resynchronize the cache by using the command `imsimta cache -sync`. Once synchronized, the channel programs will process the originally unprocessed messages after new messages are processed. If you want to change the default (4 hours), you should modify the `job_controller.cnf` file in directory `msg-srv-base/config` by adding `sync_time=timeperiod` where `timeperiod` reflects how often the queue cache is synchronized. Note that the `timeperiod` must be greater than 30 minutes. In the following example, the queue cache synchronization is modified to 2 hours by adding the `sync_time=02:00` to the global defaults section of the `job_controller.cnf`:

```
! VERSION=5.0
!IMTA job controller configuration file
!
```

```

!Global defaults
tcp_port=27442
secret=N1Y9[HzQKW
slave_command=NULL
sync_time=02:00

```

You can run `imsimta submit channel` to clear out the backlog of messages after running `imsimta cache -sync`. It is important to note that clearing out the channel may take a long time if the backlog of messages is large (greater than 1000).

For summarized queue cache information, run `imsimta qm -maint dir -database -total`.

- b. If after synchronizing the queue cache, messages are still not being delivered, you should restart the Job Controller. To do so, use the `imsimta restart job_controller` command.

Restarting the Job Controller will cause the message data structure to be rebuilt from the message queues on disk.



Caution – Restarting the Job Controller is a drastic step and should only be performed after all other avenues have been thoroughly exhausted.

Refer “[8.7 The Job Controller](#)” on page 202 for more information on the Job Controller.

- 2. Channel processing programs fail to run because they cannot create their processing log file. Check the access permissions, disk space and quotas.

26.3.8 Messages are Looping

If the MTA detects that a message is looping, that message will be sidelined as a .HELD file. See “[26.3.8.1 Diagnosing and Cleaning up .HELD Messages](#)” on page 854. Certain cases can lead to message loops which the MTA can not detect.

The first step is to determine why the messages are looping. You should look at a copy of the problem message file while it is in the MTA queue area, MTA mail log entries (if you have the logging channel keyword enabled in your MTA configuration file for the channels in question) relating to the problem message, and MTA channel debug log files for the channels in question. Determining the From: and To: addresses for the problem message, seeing the Received: header lines, and seeing the message structure (type of encapsulation of the message contents), can all help pinpoint which sort of message loop case you are encountering.

Some of the more common cases include:

- 1. A postmaster address is broken.

The MTA requires that the postmaster address be a functioning address that can receive email. If a message to the postmaster is looping, check that your configuration has a proper postmaster address pointing to an account that can receive messages.

2. Stripping of Received: header lines is preventing the MTA from detecting the message loop.

Normal detection of message loops is based on Received: header lines. If Received: header lines are being stripped (either explicitly on the MTA system itself, or on another system like a firewall), it can interfere with proper detection of message loops. In these scenarios, check that no undesired stripping of Received: header lines is occurring. Also, check for the underlying reason why the messages are looping. Possible reasons include: a problem in the assignment of system names or a system not configured to recognize a variant of its own name, a DNS problem, a lack of authoritative addressing information on the system in question, or a user address forwarding error.

3. Incorrect handling of notification messages by other messaging systems are generating reencapsulated messages in response to notification messages.

Internet standards require that notification messages (reports of messages being delivered, or messages bouncing) have an empty envelope From: address to prevent message loops. However, some messaging systems do not correctly handle such notification messages.

When forwarding or bouncing notification messages, these messaging systems may insert a new envelope From: address. This can then lead to message loops. The solution is to fix the messaging system that is incorrectly handling the notification messages.

26.3.8.1

Diagnosing and Cleaning up .HELD Messages

If the MTA detects a serious problem having to do with delivery of a message, the message is stored in a file with the suffix .HELD in */msg-svr-base/data/queue/channel*. For example:

```
% ls  
ZZ0HXZ00G0EBRBCP.HELD  
ZZ0HY200C006LGHU.HELD  
ZZ0HYA006LP6603H.HELD  
ZZ0HZ7003EOQSE37.HELD
```

.HELD files can occur due to three major reasons:

- Looping messages. The MTA detected that the messages were looping via build-up of one or another sort of Received: header lines).
- User or domain status set to hold. These are messages that are, by intent of the MTA administrator, intentionally being side-lined, typically while some maintenance procedure is being performed, (for example, while moving user mailboxes).
- Suspicious messages. Messages that met some suspicion thresh hold and were held for later manual inspection by the MTA administrator. Messages can be .HELD due to exceeding a configured maximum number of envelope recipients (see the `holdlimit` channel keyword in “[12.5.9 Expansion of Multiple Addresses](#)” on page 388), due to running the “`imsimta qc`” in *Sun Java System Messaging Server 6.3 Administration Reference*, “clean” in *Sun*

[Java System Messaging Server 6.3 Administration Reference](#) or “hold” in [Sun Java System Messaging Server 6.3 Administration Reference](#) commands based on some suspicion of the message(s) in question, or due to use of a hold action in a Sieve script.

Messages .HELD Due to Looping

Messages bouncing between servers or channels are said to be looping. Typically, a message loop occurs because each server or channel thinks the other is responsible for delivery of the message. Looping messages usually have a great many *Received: header lines. The Received: header lines will illustrate the exact path of the message loop. Look carefully at the host names and any recipient address information (for example, for recipient clauses or (ORCPT recipient) comments) appearing in such header lines. One cause of such message loops is user error.

For example, an end user may set an option to forward messages on two separate mail hosts to one another. On his `sesta.com` account, the end-user enables mail forwarding to his `varrius.com` account. And, forgetting that he has enabled this setting, he sets mail forwarding on his `varrius.com` account to his `sesta.com` account.

A loop can also occur with a faulty MTA configuration. For example, MTA Host X thinks that messages for `mail.sesta.com` go to Host Y. However, Host Y thinks that Host X should handle messages for `mail.sesta.com`; as a result, Host Y returns the mail to Host X.

In these cases, the message is ignored by the MTA and no further delivery is attempted. When such a problem occurs, look at the header lines in the message to determine which server or channel is bouncing the message. Fix the entry as needed.

Another common cause of message loops is the MTA receiving a message that was addressed to the MTA host using a network name that the MTA does not recognize (has not been configured to recognize) as one of its own names. The solution is to add the additional name to the list of names that your MTA recognizes as *its own*. Note that the MTA's thresh holds for determining that a message is looping are configurable; see the `MAX_*RECEIVED_LINES` option.dat options (“[Option File Format and Available Options](#)” in [Sun Java System Messaging Server 6.3 Administration Reference](#)). Also note that the MTA may optionally be configured--see the `HELD_SNDOPR` global MTA option--to generate a syslog notice whenever a message is forced into .HELD state due to exceeding such a thresh hold. If syslog messages of `Received count exceeded; message held.` are present, then you know that this is occurring.

You can resend the .HELD message by running “[release](#)” in [Sun Java System Messaging Server 6.3 Administration Reference](#) or following these steps:

1. Rename the .HELD extension to any 2 digit number other than 00. For example, .HELD to .06.

Note – Before renaming the .HELD file, be sure that the message has stopped looping.

2. Run `imsimta cache -sync`. Running this command will update the cache.
3. Run `imsimta submit channel` or `imsimta run channel`.

It may be necessary to perform these steps multiple times, since the message may again be marked as .HELD, because the Received: header lines accumulate. If the problem still exists, the *.HELD file will be recreated under the same channel with as before. If the problem has been addressed, the messages will be dequeued and delivered.

If you determine that the messages can simply be deleted with no attempt to deliver them, see “[clean](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* in the *Sun Java System Messaging Server 6.3 Administration Reference*.

Messages .HELD Due to User or Domain hold Status

Messages .HELD due to a user or domain status of hold--and only messages .HELD for such a reason--will normally be stored in the hold channel's queue area. That is, .HELD message files in the hold channel's queue area can be assumed to be .HELD due to user or domain status.

Messages .HELD Due to a Suspicious Characteristic

Messages .HELD due to some suspicious characteristic will of course exhibit that characteristic. The characteristic could be anything which the site has chosen to characterize as *suspicious*. MTA Administrators should stay aware of these configuration choices and actions. However, if you are not the only or original administrator of this MTA, then check the MTA configuration for any configured use of the `holdlimit` channel keyword (“[12.5.9 Expansion of Multiple Addresses](#)” on page 388), any use of the `$H` flag in address-based `*_ACCESS` mapping tables in the MTA mappings file, or any use of the `hold` action in any system Sieve file (the system level `imta.filter` file, or any channel level Sieve filters configured and named via use of `sourcefilter` or `destinationfilter` channel keywords; see “[12.12.4 Specifying Mailbox Filter File Location](#)” on page 419); and ask any fellow MTA administrators about any manual command line message holds (through, for instance, an `imsimta qm clean` command) they might have recently performed. Note also that application of a Sieve filter `hold` action, whether from a system Sieve filter or from users' personal Sieve filters, may optionally be logged; see the `LOG_FILTER` global MTA option (“[Option File Format and Available Options](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*) for more information.

26.3.9

Received Message is Encoded

Messages sent by the MTA are received in an encoded format. For example:

```
Date: Wed, 04 Jul 2001 11:59:56 -0700 (PDT)
From: "Desdemona Vilalobos" <Desdemona@sesta.com>
To: santosh@varrius.com
Subject: test message with 8bit data
MIME-Version: 1.0
Content-type: TEXT/PLAIN; CHARSET=ISO-8859-1
Content-transfer-encoding: QUOTED-PRINTABLE
```

2=00So are the Bo=F6tes Void and the Coal Sack the same?=

These messages appear unencoded when read with the MTA decoder command `imsimta decode`. Refer to the *Sun Java System Messaging Server Administration Reference* for more information.

The SMTP protocol only allows the transmission of ASCII characters (a seven-bit character set) as set forth by RFC 821. In fact, the unnegotiated transmission of eight-bit characters is illegal via SMTP, and it is known to cause a variety of problems with some SMTP servers. For example, SMTP servers can go into compute bound loops. Messages are sent over and over again. Eight-bit characters can crash SMTP servers. Finally, eight-bit character sets can wreak havoc with browsers and mailboxes that cannot handle eight-bit data.

An SMTP client used to only have three options when handling a message containing eight-bit data: return the message to the sender as undeliverable, encode the message, or send it in direct violation of RFC 821. But with the advent of MIME and the SMTP extensions, there are now standard encodings which may be used to encode eight-bit data by using the ASCII character set.

In the previous example, the recipient received an encoded message with a MIME content type of TEXT/PLAIN. The remote SMTP server (to which the MTA SMTP client transferred the message) did not support the transfer of eight-bit data. Since the original message contained eight-bit characters, the MTA had to encode the message.

26.3.10 Server-Side Rules (SSR) Are Not Working

A filter consists of one or more conditional actions to apply to a mail message. Since the filters are stored and evaluated on the server, they are often referred to as server-side rules (SSR).

This section includes information on the following SSR topics:

- “[26.3.10.1 Testing Your SSR Rules](#)” on page 858
- “[26.3.10.2 Common Syntax Problems](#)” on page 858

See also “[18.15 To Debug User-level Filters](#)” on page 574.

26.3.10.1 Testing Your SSR Rules

- To check the MTA's user filters, use the command:

```
# imsmta test -rewrite -debug -filter user@domain
```

In the output, look for the following information:

```
mmc_open_url called to open ssrf:user@ims-ms
    URL with quotes stripped: ssrd: user@ims-ms
Determined to be a SSRD URL.
    Identifier: user@ims-ms-daemon
Filter successfully obtained.
```

- In addition, you can add the slave_debug keyword to the tcp_local channel to see how a filter is applied. The results are displayed in the `tcp_local_slave.log` file. Be sure to add `mm_debug=5` in the `option.dat` file in directory `/msg-srv-base/config` in order to get sufficient debugging information.

26.3.10.2 Common Syntax Problems

- If there is a syntax problem with the filter, then look for the following message in the `tcp_local_slave.log-*` file:
`Error parsing filter expression:...`
 - If the filter is good, then filter information will be at the end of the output.
 - If the filter is bad, then the following error will be at the end of the output:
`Address list error -- 4.7.1 Filter syntax error: desdaemona@sesta.com`
- Also, if the filter is bad, then the SMTP RCPT TO command will return a temporary error response code:

```
RCPT TO: user@domain
452 4.7.1 Filter syntax error
```

26.3.11 Slow Response After Users Press Send Email Button

If users are experiencing delays when they send messages, it may be because disk input/output is reduced due to insufficiently sized message queue disks. When users press the SEND button on their email client, the MTA will not fully accept receipt of the message until the message has been committed to the message queue. Information on message queue sizing can be found

26.3.12 Asterisks in the Local Parts of Addresses or Received Fields

The MTA now checks for 8-bit characters (instead of just ASCII characters) in the local parts of addresses as well as the received fields it constructs and replaces them with asterisks.

26.4 General Error Messages

When the MTA fails to start, general error messages appear at the command line. In this section, common general error messages will be described and diagnosed.

Note – To diagnose your own MTA configuration, use the `imsimta test -rewrite -debug` utility to examine your MTA’s address rewriting and channel mapping process. By using this utility allows you to check the configuration without actually sending a message. See “[26.2.1 Check the MTA Configuration](#)” on page 836.

MTA subcomponents might also issue other error messages that are not described in this chapter. You should refer to the chapters on MTA command-line utilities and configuration in the *Sun Java System Messaging Server Administration Reference* and chapters 5 through 10 for more information on each subcomponent. This section includes the following types of errors:

- “[26.4.1 Errors in mm_init](#)” on page 859
- “[26.4.2 Compiled Configuration Version Mismatch](#)” on page 863
- “[26.4.3 Swap Space Errors](#)” on page 863
- “[26.4.4 File open or create errors](#)” on page 863
- “[26.4.5 Illegal Host/Domain Errors](#)” on page 864
- “[26.4.6 Errors in SMTP channels, os_smtp_* errors](#)” on page 865

26.4.1 Errors in mm_init

An error in `mm_init` generally indicates an MTA configuration problem. If you run the `imsimta test -rewrite` utility, these errors will be displayed. Other utilities like `imsimta cnbuild`, a channel, a server, or a browser might also return such an error.

Commonly encountered `mm_init` errors include:

- “[26.4.1.1 bad equivalence for alias...](#)” on page 860
- “[26.4.1.2 cannot open alias include file...](#)” on page 860
- “[26.4.1.3 duplicate aliases found...](#)” on page 860
- “[26.4.1.4 duplicate host in channel table...](#)” on page 860
- “[26.4.1.5 duplicate mapping name found...](#)” on page 860

- “26.4.1.6 mapping name is too long . . .” on page 861
- “26.4.1.7 error initializing ch_facility compiled character set version mismatch” on page 861
- “26.4.1.8 error initializing ch_facility no room in . . .” on page 861
- “26.4.1.9 local host alias or proper name too long for system . . .” on page 861
- “26.4.1.10 no equivalence addresses for alias . . .” on page 861
- “26.4.1.11 no official host name for channel . . .” on page 862
- “26.4.1.12 official host name is too long” on page 862

26.4.1.1

bad equivalence for alias . . .

The right hand side of an alias file entry is improperly formatted.

26.4.1.2

cannot open alias include file . . .

A file included into the alias file cannot be opened.

26.4.1.3

duplicate aliases found . . .

Two alias file entries have the same left hand side. You will need to find and eliminate the duplication. Look for an error message that says `error line #XXX` where XXX is a line number. You can fix the duplicated alias on the line.

26.4.1.4

duplicate host in channel table . . .

This error message indicates that you have two channel definitions in the MTA configuration that both have the same official host name.

Note that an extraneous blank line in the rewrite rules (upper portion) of your MTA configuration file (`imta.cnf`) causes the MTA to interpret the remainder of the configuration file as channel definitions. Make sure that the very first line of the file is not a blank. Since there are often multiple rewrite rules with the same pattern (left-hand side), this then causes MTA to interpret them as channel definitions with non-unique official host names. Check your MTA configuration for any channel definitions with duplicate official host names and for any improper blank lines in the upper (rewrite rules) portion of the file.

26.4.1.5

duplicate mapping name found . . .

This message indicates that two mapping tables have the same name, and one of the duplicate mapping tables needs to be removed. However, formatting errors in the mapping file may cause the MTA to wrongly interpret something as a mapping table name. For example, failure to properly indent a mapping table entry will cause the MTA to think that the left hand side of the entry is actually a mapping table name. Check your mapping file for general form and check the mapping table names.

Note – A blank line should precede and follow any line with a mapping table name. However, no blank lines should be interspersed among the entries of a mapping table.

26.4.1.6

mapping name is too long...

This error means that a mapping table name is too long and needs to be shortened. Formatting errors in the mapping file may cause the MTA to wrongly interpret something as a mapping table name. For example, failure to properly indent a mapping table entry will cause the MTA to think that the left hand side of the entry is actually a mapping table name. Check your mapping file and mapping table names.

26.4.1.7

error initializing ch_facility compiled character set version mismatch

If you see this message, you need to recompile and reinstall your compiled character set tables through the command `imsimta chbuild`. See the “[imsimta chbuild](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for more information.

26.4.1.8

error initializing ch_facility no room in...

This error message generally means that you need to resize your MTA character set internal tables and then rebuild the compiled character set tables with the following commands:

```
imsimta chbuild -noimage -maximum -option  
imsimta chbuild
```

Verify that nothing else needs to be recompiled or restarted before making this change. Refer to “[imsimta chbuild](#)” in *Sun Java System Messaging Server 6.3 Administration Reference* for more information on `imsimta chbuild`.

26.4.1.9

local host alias or proper name too long for system...

This error indicates that a local host alias or proper name is too long (the optional right hand side in the second or subsequent names in a channel block). However, certain syntax errors earlier in the MTA configuration file (an extraneous blank line in the rewrite rules, for instance) may cause MTA to wrongly interpret something as a channel definition. Aside from checking the indicated line of the configuration file, also check above that line for other syntax errors. In particular, if the line in which MTA issues this error is intended as a rewrite rule, then be sure to check for extraneous blank lines above it.

26.4.1.10

no equivalence addresses for alias...

An entry in the alias file is missing a right hand side (translation value).

26.4.1.11

no official host name for channel...

This error indicates that a channel definition block is missing the required second line (the official host name line). See the chapters on MTA configuration and command-line utilities in the *Sun Java System Messaging Server Administration Reference* and [Chapter 12, “Configuring Channel Definitions,”](#) for more information on channel definition blocks. A blank line is required before and after each channel definition block, but a blank line must not be present between the channel name and official host name lines of the channel definition. Also note that blank lines are not permitted in the rewrite rules portion of the MTA configuration file.

26.4.1.12

official host name is too long

The official host name for a channel (second line of the channel definition block) is limited to 128 octets in length. If you are trying to use a longer official host name on a channel, shorten it to a place holder name, and then use a rewrite rule to match the longer name to the short official host name. You may see this scenario if you work with the l (local) channel host name. For example:

Original l Channel:

```
!delivery channel to local /var/mail store
l subdirs 20 viaaliasrequired maxjobs 7 pool LOCAL_POOL
walleroo.pocofronitas.thisnameismuchtoalongandreallymakesnosensebutitisan
example.monkey.gorilla.orangutan.antidisestablishmentarianism.newt.salaman
der.lizard.gecko.komododragon.com
```

Create Place Holder:

```
!delivery channel to local /var/mail store
l subdirs 20 viaaliasrequired maxjobs 7 pool LOCAL_POOL
newt
```

Create Rewrite Rule:

```
newt.salamander.lizard.gecko.komododragon.com $U%D@newt
```

Note that when using the l (local) channel, you will need to use a REVERSE mapping table. Refer to the MTA configuration chapter in the *Sun Java System Messaging Server Administration Reference* for information on usage and syntax.

Certain syntax errors earlier in the MTA configuration file (for example, an extraneous blank line in the rewrite rules) may cause the MTA to wrongly interpret something as a channel definition. This could result in an intended rewrite rule being interpreted as an official host name. Besides checking the indicated line of the configuration file, also check above that line for other syntax errors. In particular, if the line on which the MTA issues this error is intended as a rewrite rule, be sure to check for extraneous blank lines above it.

26.4.2 Compiled Configuration Version Mismatch

One of the functions of the `imsimta cnbuild` utility is to compile MTA configuration information into an image that can be quickly loaded. The compiled format is quite rigidly defined and often changes substantially between different versions of the MTA. Minor changes might occur as part of patch releases.

When such changes occur, an internal version field is also changed so that incompatible formats can be detected. The MTA components will halt with the above error when an incompatible format is detected. The solution to this problem is to generate a new, compiled configuration with the command `imsimta cnbuild`.

It is also a good idea to use the `imsimta restart` command to restart any resident MTA server processes, so they can obtain updated configuration information.

26.4.3 Swap Space Errors

To ensure proper operation, it is important to configure enough swap space on your messaging system. The amount of required swap space will vary depending on your configuration. A general tuning recommendation is that the amount of swap space should be at least three times the amount of main memory.

An error message such as the following indicates a lack of swap space:

```
jbc_channels: chan_execute [1]: fork failed: Not enough space
```

You might see this error in the Job Controller log file. Other swap space errors will vary depending on your configuration.

Use the following commands to determine how much swap space you have left and determine how much you have used:

- Solaris systems: `swap -s` (at the time MTA processes are busy), `ps -elf`, or `tail /var/adm/messages`
- HP-UX systems: `swapon` or `tail /var/adm/syslog/syslog.log`

26.4.4 File open or create errors

In order to send a message, the MTA reads configuration files and creates message files in the MTA message queue directories. Configuration files must be readable by the MTA or any program written against the MTA's SDKs. During installation, proper permissions are assigned to these files. The MTA utilities and procedures which create configuration files also assign permissions. If the files are protected by the system manager, other privileged user, or through some site-specific procedure, the MTA may not be able to read configuration information. This

will result in “File open” errors or unpredictable behavior. The `imsimta test - rewrite` utility reports additional information when it encounters problems reading configuration files. See “[imsimta test](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

If the MTA appears to function from privileged accounts but not from unprivileged accounts, then file permissions in the MTA table directory are likely the cause of the problem. Check the permissions on configuration files and their directories. See “[26.2.3 Check the Ownership of Critical Files](#)” on page [837](#).

“File create” errors usually indicate a problem while creating a message file in an MTA message queue directory. See “[26.2.2 Check the Message Queue Directories](#)” on page [836](#) to diagnose file creation problems.

26.4.5 Illegal Host/Domain Errors

You may see this error when an address is provided to the MTA through a browser. Or, the error may be deferred and returned as part of an error return mail message. In both cases, this error message indicates that the MTA is not able to deliver mail to the specified host. To determine why the mail is not being sent to the specified host, you should follow these troubleshooting procedures:

- Verify that the address in question is not misspelled, is not transcribed incorrectly, or does not use the name of a host or domain that no longer exists.
- Run the address in question through the `imsimta test - rewrite` utility. If this utility also returns an “illegal host/domain” error on the address, then MTA has no rules in the `imta.cnf` file and related files to handle the address. Verify that you have configured MTA correctly, that you answered all configuration questions appropriately, and that you have kept your configuration information up to date.
- If `imsimta test - rewrite` does not encounter an error on the address, then MTA is able to determine how to handle the address, but the network transport will not accept it. You can examine the appropriate log files from the delivery attempt for additional details. Transient network routing or name service errors should not result in returned error messages, though it is possible for badly misconfigured domain name servers to cause these problems.
- If you are on the Internet, check that you have properly configured your TCP/IP channel to support MX record lookups. Many domain addresses are not directly accessible on the Internet and require that your mail system correctly resolve MX entries. If you are on the Internet and your TCP/IP is configured to support MX records, you should have configured the MTA to enable MX support; see TCP/IP Connection and DNS Lookup Support “[12.4.3 TCP/IP Connection and DNS Lookup Support](#)” on page [368](#) for more information. If your TCP/IP package is not configured to support MX record lookups, then you will not be able to reach MX-only domains.

26.4.6 Errors in SMTP channels, os_smtp_* errors

Errors such as the following are not necessarily MTA errors: `os_smtp_*` errors like `os_smtp_open`, `os_smtp_read`, and `os_smtp_write` errors. These errors are generated when the MTA reports a problem encountered at the network layer. For example, an `os_smtp_open` error means that the network connection to the remote side could not be opened. The MTA may be configured to connect to an invalid system because of addressing errors or channel configuration errors. The `os_smtp_*` errors are commonly due to DNS or network connectivity problems, particularly if this was a previously working channel or address. `os_smtp_read` or `os_smtp_write` errors are usually an indication that the connection was aborted by the other side or due to network problems.

Network and DNS problems are often transient in nature. The occasional `os_smtp_*` error is usually nothing to be concerned about. However, if you are consistently seeing these errors, it may be an indication of an underlying network problem.

To obtain more information about a particular `os_smtp_*` error, enable debugging on the channel in question. Investigate the debug channel log file that will show details of the attempted SMTP dialogue. In particular, look at the timing of when a network problem occurred during the SMTP dialogue. The timing may suggest the type of network or remote side issue. In some cases, you may also want to perform network level debugging (for example, TCP/IP packet tracing) to determine what was sent or received.

Monitoring Messaging Server

In most cases, a well-planned, well-configured server will perform without extensive intervention from an administrator. As an administrator, however, it is your job to monitor the server for signs of problems. This chapter describes the monitoring of the Messaging Server. It consists of the following sections:

- “[27.1 Automatic Monitoring and Restart](#)” on page 867
- “[27.2 Daily Monitoring Tasks](#)” on page 868
- “[27.3 Monitoring System Performance](#)” on page 869
- “[27.4 Monitoring the MTA](#)” on page 872
- “[27.5 Monitoring LDAP Directory Server](#)” on page 875
- “[27.6 Monitoring Message Access](#)” on page 875
- “[27.7 Monitoring the Message Store](#)” on page 877
- “[27.8 Utilities and Tools for Monitoring](#)” on page 878

Troubleshooting procedures can be found in [Chapter 26, “Troubleshooting the MTA”](#)

27.1 Automatic Monitoring and Restart

Messaging Server provides a way to transparently monitor services and automatically restart them if they crash or become unresponsive (the services hangs or freeze up). It can monitor all message store, MTA, and MMP services including the IMAP, POP, HTTP, job controller, dispatcher, and MMP servers. It does not monitor other services such as SMS or TCP/SNMP servers. (TCP/SNMP is monitored by the job controller.) Refer to “[4.5 Automatic Restart of Failed or Unresponsive Services](#)” on page 127 and “[27.8.9 Monitoring Using msprobe and watcher Functions](#)” on page 886.

27.2 Daily Monitoring Tasks

The most important tasks you should perform on a daily basis are checking postmaster mail, monitoring the log files, and setting up the `stored` utility. These tasks are described below.

27.2.1 Checking postmaster Mail

Messaging Server has a predefined administrative mailing list set up for postmaster email. Any users who are part of this mailing list will automatically receive mail addressed to postmaster.

The rules for postmaster mail are defined in RFC822, which requires every email site to accept mail addressed to a user or mailing list named postmaster and that mail sent to this address be delivered to a real person. All messages sent to `postmaster@host.domain` are sent to a postmaster account or mailing list.

Typically, the postmaster address is where users should send email about their mail service. As postmaster, you might receive mail from local users about server response time, from other server administrators who are encountering problems sending mail to your server, and so on. You should check postmaster mail daily.

You can also configure the server to send certain error messages to the postmaster address. For example, when the MTA cannot route or deliver a message, you can be notified via email sent to the postmaster address. You can also send exception condition warnings (low disk space, poor server response) to postmaster.

27.2.2 Monitoring and Maintaining the Log Files

Messaging Server creates a separate set of log files for each of the major protocols or services it supports including SMTP, IMAP, POP, and HTTP. These are located in `msg-svr-base/data/log`. You should monitor the log files on a routine basis--especially if you are having problems with the server.

Be aware that logging can impact server performance. The more verbose the logging you specify, the more disk space your log files will occupy for a given amount of time. You should define effective but realistic log rotation, expiration, and backup policies for your server. For information about defining logging policies for your server, see [Chapter 25, “Managing Logging”](#).

27.2.3 Setting Up the msprobe Utility

The `msprobe` utility automatically performs monitoring and restart functions. For further information see [“27.8.9 Monitoring Using msprobe and watcher Functions”](#) on page 886

27.3 Monitoring System Performance

This chapter focuses on Messaging Server monitoring, however, you will also need to monitor the system on which the server resides. A well-configured server cannot perform well on a poorly-tuned system, and symptoms of server failure may be an indication that the hardware is not powerful enough to serve the email load. This chapter does not provide all the details for monitoring system performance as many of these procedures are platform specific and may require that you refer to the platform specific system documentation. The following procedures are described here for performance monitoring:

- “[27.3.1 Monitoring End-to-end Message Delivery Times](#)” on page 869
- “[27.3.2 Monitoring Disk Space](#)” on page 869
- “[27.3.3 Monitoring CPU Usage](#)” on page 872

27.3.1 Monitoring End-to-end Message Delivery Times

Email needs to be delivered on time. This may be a service agreement requirement, but also it is good policy to have mail delivered as quickly as possible. Slow end-to-end times could indicate a number of things. It may be that the server is not working properly, or that certain times of the day experience overwhelming message loads, or that the existing hardware resources are being pushed beyond their capacity.

27.3.1.1 Symptoms of Poor End-to-end Message Delivery Times

Mail takes a longer period of time to be delivered than normal.

27.3.1.2 To Monitor End-to-end Message Delivery Times

- Use any facility that sends a message and receives it. Compare the headers times between server hops, and times between point of origin and retrieval. See “[27.8.1 immonitor-access](#)” on page 878.

27.3.2 Monitoring Disk Space

Inadequate disk space is one of the most common causes of the mail server problems and failure. Without space to write to the MTA queues or to the message store, the mail server will fail. In addition, unless log files are monitored and cleaned up, they can grow uncontrollably filling up all disk space.

Message store partitions grow as new messages are delivered to the mailboxes; for example, if message store quotas are not enforced, the message store can outgrow the disk space available for a partition. Another cause of running out of disk space are the MTA message queues growing too large. A third area of concern is if a problem occurs with the log file monitoring

facilities and the log files growing uncontrollably. (Note that there are a number of log files such as LDAP, MTA, and Message Access, and that each of these log files can be stored on different disks.)

27.3.2.1

Symptoms of Disk Space Problems

Different symptoms can occur depending on which disk or partition is running out of space. MTA queues can overflow and reject SMTP connections, messages might remain in the ims_master queue and not be delivered to the message store, and log files can overflow.

If a message store partition fills up, message access daemons can fail, and message store data can be corrupted. Message store maintenance utilities such as `imexpire` and `reconstruct` can repair the damage and reduce disk usage. However, these utilities require additional disk space, and repairing a partition that has filled an entire disk can potentially cause down time.

27.3.2.2

To Monitor Disk Space

Depending upon the system configuration you may need to monitor various disks and partitions. For example, MTA queues may reside on one disk/partition, message stores may reside on another, and log files may reside on yet another. Each of these spaces will require monitoring and the methods to monitor these spaces may differ.

Messaging Server provides specific methods for monitoring message store disk usage and preventing partitions from filling up all available disk space.

You can take the following steps to monitor the message store's use of disk space:

- Set parameters to monitor message store disk usage
- Lock message store partitions when a disk-usage threshold is reached

For details, see the sections that follow: “[Monitoring the Message Store](#)” on page 870 and “[Monitoring Message Store Partitions](#)” on page 871.

Monitoring the Message Store

It is recommended that message store disk usage not exceed 75% capacity. You can monitor message store disk usage by configuring the following alarm attributes using the `configutil` utility:

- `alarm.diskavail.msgalarmstatinterval`
- `alarm.diskavail.msgalarmthreshold`
- `alarm.diskavail.msgalarmwarninginterval`
- `alarm.diskavail.msgalarmdescription`

By setting these parameters, you can specify how often the system should monitor disk space and under what circumstances the system should send a warning. For example, if you want the system to monitor disk space every 600 seconds, specify the following command:

```
configutil -o alarm.diskavail.msgalarmstatinterval -v 600
```

If you want to receive a warning whenever available disk space falls below 20%, specify the following command:

```
configutil -o alarm.diskavail.msgalarmthreshold -v 20
```

Refer to [Table 27–6](#) for more information on these parameters.

Monitoring Message Store Partitions

You can halt messages from being delivered to a message store partition when the partition fills more than a specified percentage of available disk space. This is done by setting two configutil parameters to enable the feature and specify the disk-usage threshold.

With this feature, the message store daemon monitors the partition's disk usage. As disk usage increases, the store daemon dynamically checks the partition more frequently (ranging from once every 100 minutes to once a minute).

If disk usage goes higher than the specified threshold, the store daemon:

- Locks the partition. Incoming messages are held in the MTA message queue, but not delivered to the mailboxes in the message store partition.
- Logs a message to the default log file.
- Sends an email notification to the postmaster. (You can change the recipient of the email by setting the configutil parameter `alarm.msgalarmnoticercpt`.)

When disk usage falls below the threshold, the partition is unlocked, and messages are again delivered to the store.

The configutil parameters are as follows:

- `local.store.checkdiskusage` enables the partition-monitoring feature.
Allowable values: yes, no
Default value: yes
- `local.store.diskusagethreshold` specifies the disk-usage threshold. The value of `local.store.diskusagethreshold` is a percentage from 1 to 99.
Default value: 99

You should set the disk-usage threshold to a percentage low enough to give you time to repartition or assign more disk space to the local message store.

For example, suppose a partition fills up disk space at a rate of 2 percent per hour, and it takes an hour to allocate additional disk space for the local message store. In this case, you should set the disk-usage threshold to a value lower than 98 percent.

Monitoring the MTA Queues and Logging Space

You will need to monitor MTA queue disk and logging space disk usage.

For information on managing logging space, see Chapter 25, “Managing Logging.” For example, to learn how to monitor the `mail.log` file, see “[25.3 Managing MTA Message and Connection Logs](#)” on page 796

27.3.3

Monitoring CPU Usage

High CPU usage is either a sign that there is not enough CPU capacity for the level of usage or some process is using up more CPU cycles than is appropriate.

27.3.3.1

Symptoms of CPU Usage Problems

Poor system response time. Slow logging in of users. Slow rate of delivery.

27.3.3.2

To Monitor CPU Usage

Monitoring CPU usage is a platform specific task. Refer to the relevant platform documentation.

27.4 Monitoring the MTA

This section consists of the following subsections:

- [“27.4.1 Monitoring the Size of the Message Queues” on page 872](#)
- [“27.4.2 Monitoring Rate of Delivery Failure” on page 873](#)
- [“27.4.3 Monitoring Inbound SMTP Connections” on page 873](#)
- [“27.4.4 Monitoring the Dispatcher and Job Controller Processes” on page 874](#)

27.4.1

Monitoring the Size of the Message Queues

Excessive message queue growth may indicate that messages are not being delivered, are being delayed in their delivery, or are coming in faster than the system can deliver them. This may be caused by a number of reasons such as a denial of service attack caused by huge numbers of messages flooding your system, or the Job Controller not running.

See [“8.5.2 Channel Message Queues” on page 199](#), [“26.3.6 Messages are Not Dequeued” on page 849](#) and [“26.3.7 MTA Messages are Not Delivered” on page 852](#) for more information on message queues.

27.4.1.1 Symptoms of Message Queue Problems

- Disk space usage grows.
- User not receiving messages in a reasonable time.
- Message queue sizes are abnormally high.

27.4.1.2 To Monitor the Size of the Message Queues

Probably the best way to monitor the message queues is to use `imsimta qm` and `imsimta summarize`. Refer to “[27.8.6 imsimta qm counters](#)” on page [885](#).

You can also monitor the number of files in the queue directories (`msg-svr-base/data/queue/`). The number of files will be site-specific, and you’ll need to build a baseline history to find out what is “too many.” This can be done by recording the size of the queue files over a two week period to get an approximate average.

27.4.2 Monitoring Rate of Delivery Failure

A delivery failure is a failed attempt to deliver a message to an external site. A large increase in rate of delivery failure can be a sign of a network problem such as a dead DNS server or a remote server timing out on responding to connections.

27.4.2.1 Symptoms of Rate of Delivery Failure

There are no outward symptoms. Lots of Q records will appear in to `mail.log_current`.

27.4.2.2 To Monitor the Rate of Delivery Failure

Delivery failures are recorded in the MTA logs with the logging entry code Q. Look at the record in the file `msg-svr-base/data/log/mail.log_current`. Example:

```
mail.log:06-Oct-2003 00:24:03.66 501d.0b.9 ims-ms Q 5 durai.balusamy@Sun.COM
rfc822;durai.balusamy@Sun.COM durai@ims-ms-daemon
<00ce01c38bda$c7e2b240$6501a8c0@guindy> Mailbox is busy
```

27.4.3 Monitoring Inbound SMTP Connections

An unusual increase in the number of inbound SMTP connections from a given IP address may indicate:

- An external user is trying to relay mail.
- An external user is trying to do a service denial attack.

27.4.3.1 Symptoms of Unauthorized SMTP Connections

- **External user relaying mail:** No outward symptoms.

- **Service denial attack:** External attempt to overload the SMTP servers with message requests.

27.4.3.2

To Monitor Inbound SMTP Connections

- **External user relaying mail:** Look in *msg-svr-base/log/mail.log_current* for records with the logging entry code J (rejected relays). To turn on logging of remote IP addresses add the following line to the *option.dat* file:

```
log_connection=1
```

Note that there is a slight performance trade-off when this feature is enabled.

- **Service denial attack:** To find out who and how many users are connecting to the SMTP servers, you can run the command *netstat* and check for connections at the SMTP port (default: 25). Example:

Local address	Remote address					State
192.18.79.44.25	192.18.78.44.56035	32768	0	32768	0	CLOSE_WAIT
192.18.79.44.25	192.18.136.54.57390	8760	0	24820	0	ESTABLISHED
192.18.79.44.25	192.18.26.165.48508	33580	0	24820	0	TIME_WAIT

Note that you will first need to determine the appropriate number of SMTP connections and their states (ESTABLISHED, CLOSE_WAIT, etc.) for your system to determine if a particular reading is out of the ordinary.

If you find many connections staying in the SYN_RECEIVED state this might be caused by a broken network or a denial of service attack. In addition, the lifetime of an SMTP server process is limited. This is controlled by the MTA configuration variable MAX_LIFE_TIME in the *dispatcher.cnf* file. The default is 86,400 seconds (one day). Similarly, MAX_LIFE_CONNS specifies the maximum number of connections a server process can handle in its lifetime. If you find a particular SMTP server that has around for a long time you may wish to investigate.

27.4.4

Monitoring the Dispatcher and Job Controller Processes

The Dispatcher and Job Controller Processes must be operating for MTA to work. You should have one process of each kind.

27.4.4.1

Symptoms of Dispatcher and Job Controller Processes Down

If the Dispatcher is down or does not have enough resources, SMTP connections are refused.

If the Job Controller is down, queue size will grow.

27.4.4.2 To Monitor Dispatcher and Job Controller Processes

Check to see that the processes called `dispatcher` and `job_controller` exist. See “[26.2.4 Check that the Job Controller and Dispatcher are Running](#)” on page 837.

27.5 Monitoring LDAP Directory Server

This section consists of the following subsection:

- [“27.5.1 Monitoring slapd” on page 875](#)

27.5.1 Monitoring slapd

The LDAP directory server (`slapd`) provides directory information for the messaging system. If `slapd` is down, the system will not work properly. If `slapd` response time is too slow, this will affect login speed and any other transaction that requires LDAP lookups.

27.5.1.1 Symptoms of slapd Problems

- Client POP, IMAP, or Webmail Authentication fails or slower than expected.
- MTA not working properly

27.5.1.2 To Monitor slapd

- Check that `ns-slapd` process is running.
- Check `slapd` log files access and errors in `slapd-instance/logs/`.
- Check the `ns-slapd` response time while searching for a user.
- See also “[27.8.1 immonitor-access](#)” on page 878

27.6 Monitoring Message Access

This section consists of the following subsections:

- [“27.6.1 Monitoring imapd, popd and httpd” on page 875](#)
- [“27.7.1 Monitoring stored” on page 877](#)

27.6.1 Monitoring imapd, popd and httpd

These processes provide access to IMAP, POP and Webmail services. If any of these is not running or not responding, the service will not function appropriately. If the service is running, but is over loaded, monitoring will allow you to detect this and configure it more appropriately.

27.6.1.1

Symptoms of imapd, popd and httpd Problems

Connections are refused or system is too slow to connect. For example, if IMAP is not running and you try to connect to IMAP directly you will see something like this:

```
telnet 0 143 Trying 0.0.0.0... telnet: Unable to connect to remote host:  
Connection refused
```

If you try to connect with a client, you will get a message such as:

“Client is unable to connect to the server at the location you have specified. The server may be down or busy.”

27.6.1.2

To Monitor imapd, popd and httpd

- Can be monitored with `watcher` and `msprobe`. See “[4.5 Automatic Restart of Failed or Unresponsive Services](#)” on page [127](#) and “[27.8.9 Monitoring Using msprobe and watcher Functions](#)” on page [886](#)
- Can be monitored with SNMP.

If you have the SNMP set up, this is a very good way to monitor these processes. See [Appendix A, “SNMP Support.”](#) The server information is in the Network Services Monitoring MIB.

- Check log files.

Look in the directory `msg-svr-base/log/service` where `service` can be `http` or IMAP or POP. In that directory you will find a number of log files. One filename is the name of the `service` (`imap`, `pop`, `http`) and the others are the name of the service plus a sequence number and a date concatenated to the service name. For example:

```
imap imap.29.1010221593 imap.31.1010394412 imap.33.1010567224
```

The file with just the service name is the latest log. The other ones are ordered by the sequence number (here 29, 31, 33) and the one with the highest sequence number is the next newest one. (See [Chapter 25, “Managing Logging.”](#))

If a server was shut down you might see something like this:

```
imap.12.1065431243:[07/Oct/2003:01:15:43 -0700] gotmail-2 imapd[20525]: General  
Warning: Sun Java System Messaging Server IMAP4 6.1 (built Sep 24 2003) shutting down
```

- Can be checked with `counterutil`. See “[27.8.3 counterutil](#)” on page [879](#) and “[counterutil](#)” in [Sun Java System Messaging Server 6.3 Administration Reference](#).
- Run the platform-specific command to verify that the `imapd`, `popd` and `httpd` processes are running. For example, in Solaris you can use the `ps` command and look for `imapd`, `popd` and `mshttpd`.
- You can set alarms for specified server performance thresholds by setting the server response configuration parameters described in “[27.8.9.1 Alarm Messages](#)” on page [888](#)
- See “[27.8.1 immonitor-access](#)” on page [878](#).

27.7 Monitoring the Message Store

Messages are stored in a database. The distribution of users on disks, the size of their mailbox, and disk requirements affect the store performance. These are described in the following subsections:

- “27.7.1 Monitoring stored” on page 877
- “27.7.2 Monitoring the State of Message Store Database Locks” on page 878

27.7.1 Monitoring stored

`stored` performs a variety of important tasks such as deadlock and transaction operations of the message database, enforcing aging policies, and expunging and erasing messages stored on disk. If `stored` stops running, the messaging server will eventually run into problems. If `stored` doesn't start when `start-msg` is run, no other processes will start. For more information about `stored` see “`stored`” in *Sun Java System Messaging Server 6.3 Administration Reference*.

27.7.1.1 Symptoms of stored Problems

There are no outward symptoms.

27.7.1.2 To Monitor stored

- Check that the `stored` process is running. `stored` creates and updates a `pid` file in `msg-svr-base/data/proc` called `store`. The `pid` file shows an `init` state when recovering and a `ready` state when ready. For example:

```
231: cat store
28250
ready
```

The number on the first line is the process ID of `stored`.

```
232: ps -eaf | grep stored
inetuser 28250 1 0 Jan 05 ? 8:44
/opt/SUNWmsgsr/lib/stored -d
```

- Check for log file build up in `msg-svr-base/store/mboxlist`. Note that not every log file build up is caused by direct `stored` problems. Log files may also build up if `imapd` dies or there is a database problem.
- Check the timestamp on the following files in `msg-svr-base/config`:
 - `stored.ckpt` - Touched when attempt at checkpointing is made. Should get time stamped every 1 minute
 - `stored.lcu` - Touched at every db log cleanup. Should get time stamped every 5 minutes
 - `stored.per` - Touched at every spawn of peruser db writeout. Should get time stamped every 60 minutes

- Check for stored messages in the default log file *msg-svr-base/log/default/default*
- Can be monitored with `watcher` and `msprobe`. See “[4.5 Automatic Restart of Failed or Unresponsive Services](#)” on page 127 and “[27.8.9 Monitoring Using msprobe and watcher Functions](#)” on page 886.

27.7.2 Monitoring the State of Message Store Database Locks

The state of database-locks is held by different server processes. These database locks can affect the performance of the message store. In case of deadlocks, messages will not be getting inserted into the store at reasonable speeds and the `ims-ms` channel queue will grow larger as a result. There are legitimate reasons for a queue to back up, so it is useful to have a history of the queue length in order to diagnose problems.

27.7.2.1 Symptoms of Message Store Database Lock Problems

Number of transactions are accumulating and not resolving.

27.7.2.2 To Monitor Message Store Database Locks

Use the command `imcheck -s` (used to be `counterutil -o db_lock`)

27.8 Utilities and Tools for Monitoring

The following tools are available in for monitoring:

- “[27.8.1 immonitor-access](#)” on page 878
- “[27.8.2 imcheck](#)” on page 879
- “[27.8.3 counterutil](#)” on page 879
- “[27.8.4 Log Files](#)” on page 882
- “[27.8.5 imsmta counters](#)” on page 882
- “[27.8.6 imsmta qm counters](#)” on page 885
- “[27.8.7 MTA Monitoring Using SNMP](#)” on page 885
- “[27.8.8 imquotacheck for Mailbox Quota Checking](#)” on page 886
- “[27.8.9 Monitoring Using msprobe and watcher Functions](#)” on page 886

27.8.1 immonitor-access

`immonitor-access` monitors the status of the following Messaging Server components/processes: Mail Delivery (SMTP server), Message Access and Store (POP and IMAP servers), Directory Service (LDAP server) and HTTP server. This utility measures the

response times of the various services and the total round trip time taken to send and retrieve a message. The Directory Service is monitored by looking up a specified user in the directory and measuring the response time. Mail Delivery is monitored by sending a message (SMTP) and the Message Access and Store is monitored by retrieving it. Monitoring the HTTP server is limited to finding out whether or not it is up and running.

For complete instructions, refer to “[immonitor-access](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

27.8.2 imcheck

Use `imcheck -s` to monitor database statistics including logs and transactions.

27.8.3 counterutil

This utility provides statistics acquired from different system counters. Here is a current list of available counter objects:

```
# /opt/SUNWmsgsr/sbin/counterutil -l
Listing registry (/opt/SUNWmsgsr/data/counter/counter)
numobjects = 11
refcount = 1
created = 25/Sep/2003:02:04:55 -0700
modified = 02/Oct/2003:22:48:55 -0700
    entry = alarm
    entry = diskusage
    entry = serverresponse      entry = imapstat
    entry = httpstat
    entry = popstat
    entry = cgimsg
```

Each entry represents a counter object and supplies a variety of useful counts for this object. In this section we will only be discussing the `alarm`, `diskusage`, `serverresponse`, `popstat`, `imapstat`, and `httpstat` counter objects. For details on `counterutil` command usage, refer to “[counterutil](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

27.8.3.1 counterutil Output

`counterutil` has a variety of flags. A command format for this utility may be as follows:

```
counterutil -o CounterObject -i 5 -n 10
```

where,

`-o CounterObject` represents the counter object `alarm`, `diskusage`, `serverresponse`, `popstat`, `imapstat`, and `httpstat`.

- i 5 specifies a 5 second interval.
- n 10 represents the number of iterations (default: infinity).

An example of `counterutil` usage is as follows:

```
# counterutil -o imapstat -i 5 -n 10
Monitor counterobject (imapstat)
registry /gotmail/iplanet/server5/msg-gotmail/counter/counter opened
counterobject imapstat opened

count = 1 at 972082466 rh = 0xc0990 oh = 0xc0968

global.currentStartTime [4 bytes]: 17/Oct/2000:12:44:23 -0700
global.lastConnectionTime [4 bytes]: 20/Oct/2000:15:53:37 -0700
global.maxConnections [4 bytes]: 69
global.numConnections [4 bytes]: 12480
global.numCurrentConnections [4 bytes]: 48
global.numFailedConnections [4 bytes]: 0
global.numFailedLogins [4 bytes]: 15
global.numGoodLogins [4 bytes]: 10446
...
...
```

27.8.3.2

Alarm Statistics Using `counterutil`

These alarm statistics refer to the alarms sent by `stored`. The alarm counter provides the following statistics:

TABLE 27-1 `counterutil` alarm Statistics

Suffix	Description
alarm.countoverthreshold	Number of times crossing threshold.
alarm.countwarningsent	Number of warnings sent.
alarm.current	Current monitored valued.
alarm.high	Highest ever recorded value.
alarm.low	Lowest ever recorded value.
alarm.timelastset	The last time current value was set.
alarm.timelastwarning	The last time warning was sent.
alarm.timereset	The last time reset was performed.
alarm.timestatechanged	The last time alarm state changed.

TABLE 27-1 counterutil alarm Statistics *(Continued)*

Suffix	Description
alarm.warningstate	Warning state (yes(1) or no(0)).

27.8.3.3**IMAP, POP, and HTTP Connection Statistics Using counterutil**

To get information on the number of current IMAP, POP, and HTTP connections, number of failed logins, total connections from the start time, and so forth, you can use the command `counterutil -o CounterObject -i 5 -n 10`. where *CounterObject* represents the counter object `popstat`, `imapstat`, or `httpstat`. The meaning of the `imapstat` suffixes is shown in [Table 27-2](#). The `popstat` and `httpstat` objects provide the same information in the same format and structure.

TABLE 27-2 counterutil imapstat Statistics

Suffix	Description
<code>currentStartTime</code>	Start time of the current IMAP server process.
<code>lastConnectionTime</code>	The last time a new client was accepted.
<code>maxConnections</code>	Maximum number of concurrent connections handled by IMAP server.
<code>numConnections</code>	Total number of connections served by the current IMAP server.
<code>numCurrentConnections</code>	Current number of active connections.
<code>numFailedConnections</code>	Number of failed connections served by the current IMAP server.
<code>numFailedLogins</code>	Number of failed logins served by the current IMAP server.
<code>numGoodLogins</code>	Number of successful logins served by the current IMAP server.

27.8.3.4**Disk Usage Statistics Using counterutil**

The command: `counterutil -o diskusage` generates following information:

TABLE 27-3 counterutil diskstat Statistics

Suffix	Description
<code>diskusage.availSpace</code>	Total space available in the disk partition.
<code>diskusage.lastStatTime</code>	The last time statistic was taken.
<code>diskusage.mailPartitionPath</code>	Mail partition path.
<code>diskusage.percentAvail</code>	Disk partition space available percentage.
<code>diskusage.totalSpace</code>	Total space in the disk partition.

27.8.3.5 Server Response Statistics

The command: `counterutil -o serverresponse` generates following information. This information is useful for checking if the servers are running, and how quickly they're responding.

TABLE 27-4 `counterutil serverresponse` Statistics

Suffix	Description
<code>http.laststattime</code>	Last time http server response was checked.
<code>http.responsetime</code>	Response time for the http.
<code>imap.laststattime</code>	Last time imap server response was checked.
<code>imap.responsetime</code>	Response time for the imap.
<code>pop.laststattime</code>	Last time pop server response was checked.
<code>pop.responsetime</code>	Response time for the pop.

27.8.4 Log Files

Messaging server logs event records for SMTP, IMAP, POP, and HTTP. The policies for creating and managing the Messaging Server log files are customizable.

Since logging can affect the server performance, logging should be considered very carefully before the burden is put on the server. Refer to [Chapter 25, “Managing Logging,”](#) for more information.

27.8.5 `imsimta` counters

The MTA accumulates message traffic counters based upon the Mail Monitoring MIB, RFC 1566 for each of its active channels. The channel counters are intended to help indicate the trend and health of your e-mail system. Channel counters are not designed to provide an accurate accounting of message traffic. For precise accounting, instead see MTA logging as discussed in [Chapter 25, “Managing Logging.”](#)

The MTA channel counters are implemented using the lightest weight mechanisms available so that they cause as little impact as possible on actual operation. Channel counters do not try harder: if an attempt to map the section fails, no information is recorded; if one of the locks in the section cannot be obtained almost immediately, no information is recorded; when a system is shut down, the information contained in the in-memory section is lost forever.

The `imsimta counters -show` command provides MTA channel message statistics (see below). These counters need to be examined over time noting the minimum values seen. The minimums may actually be negative for some channels. A negative value means that there were

messages queued for a channel at the time that its counters were zeroed (for example, the cluster-wide database of counters created). When those messages were dequeued, the associated counters for the channel were decremented and therefore leading to a negative minimum. For such a counter, the correct “absolute” value is the current value less the minimum value that counter has ever held since being initialized.

Channel	Messages	Recipients	Blocks	
tcp_local				
Received	29379	79714	982252	(1)
Stored	61	113	-2004	(2)
Delivered	29369	79723	983903 (29369 first time)	(3)
Submitted	13698	13699	18261	(4)
Attempted	0	0	0	(5)
Rejected	1	10	0	(6)
Failed	104	104	4681	(7)
Queue time/count		16425/29440 = 0.56		(8)
Queue first time/count		16425/29440 = 0.56		(9)
Total In Assocs		297637		
Total Out Assocs		28306		

1) Received is the number of messages enqueued to the channel named `tcp_local`. That is, the messages enqueued (E records in the `mail.log*` file) to the `tcp_local` channel by any other channel.

2) Stored is the number of messages stored in the channel queue to be delivered.

3) Delivered is the number of messages which have been processed (dequeued) by the channel `tcp_local`. (That is, D records in the `mail.log*` file.) A dequeue operation may either correspond to a successful delivery (that is, an enqueue to another channel), or to a dequeue due to the message being returned to the sender. This will generally correspond to the number Received minus the number Stored.

The MTA also keeps track of how many of the messages were dequeued upon first attempt; this number is shown in parentheses.

4) Submitted is the number of messages enqueued (E records in the `mail.log` file) by the channel `tcp_local` to any other channel.

5) Attempted is the number of messages which have experienced temporary problems in dequeuing, that is, Q or Z records in the `mail.log*` file.

6) Rejected is the number of attempted enqueues which have been rejected, that is, J records in the `mail.log*` file.

7) Failed is the number of attempted dequeues which have failed, that is, R records in the `mail.log*` file.

8) Queue time/count is the average time-spent-in-queue for the delivered messages. This includes both the messages delivered upon the first attempt, see (9), and the messages that required additional delivery attempts (hence typically spent noticeable time waiting fallow in the queue).

9) Queue first time/count is the average time-spent-in-queue for the messages delivered upon the first attempt.

Note that the number of messages submitted can be greater than the number delivered. This is often the case, since each message the channel dequeues (delivers) will result in at least one new message enqueued (submitted) but possibly more than one. For example, if a message has two recipients reached via different channels, then two enqueues will be required. Or if a message bounces, a copy will go back to the sender and another copy may be sent to the postmaster. Usually that will be two submissions (unless both are reached through the same channel).

More generally, the connection between Submitted and Delivered varies according to type of channel. For example, in the conversion channel, a message would be enqueued by some other arbitrary channel, and then the conversion channel would process that message and enqueue it to a third channel and mark the message as dequeued from its own queue. Each individual message takes a path:

```
elsewhere -> conversion E record Received  
conversion -> elsewhere E record Submitted  
conversion           D record Delivered
```

However, for a channel such as `tcp_local` which is not a “pass through,” but rather has two separate pieces (slave and master), there is no connection between Submitted and Delivered. The Submitted counter has to do with the SMTP server portion of the `tcp_local` channel, whereas the Delivered counter has to do with the SMTP client portion of the `tcp_local` channel. Those are two completely separate programs, and the messages travelling through them may be completely separate.

Messages submitted to the SMTP server:

```
tcp_local -> elsewhere E record Submitted
```

Messages sent out to other SMTP hosts via the SMTP client:

```
elsewhere -> tcp_local E record Received  
tcp_local           D record Delivered
```

Channel dequeues (delivers) will result in at least one new message enqueued (submitted) but possibly more than one. For example, if a message has two recipients reached via different channels, then two enqueues will be required. Or if a message bounces, a copy will go back to the sender and another copy may be sent to the postmaster. Usually that will be reached through the same channel.

27.8.5.1 Implementation on UNIX and NT

For performance reasons, a node running the MTA keeps a cache of channel counters in memory using a shared memory section (UNIX) or shared file-mapping object (NT). As processes on the node enqueue and dequeue messages, they update the counters in this in-memory cache. If the in-memory section does not exist when a channel runs, the section will be created automatically. (The `imta start` command also creates the in-memory section, if it does not exist.)

The command `imta counters -clear` or the `imta qm` command `counters clear` may be used to reset the counters to zero.

27.8.6 imsimta qm counters

The `imsimta qm` utility displays MTA channel queue message counters. You must be root or `mailsrv` to run this utility. The output fields are the same as those described in “[27.8.5 imsimta counters](#)” on page 882. See also “[imsimta counters](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

Example:

```
# imsimta counters -create
# imsimta qm counters show
Channel          Messages   Recipients  Blocks
-----
tcp_intranet
  Received      13077     13859    264616
  Stored         92        91       -362
  Delivered     12985     13768    264978
  Submitted      2594      2594     3641
...

```

Every time you restart the MTA, you must run: `# imsimta counters -create`

27.8.7 MTA Monitoring Using SNMP

Messaging Server supports system monitoring through the Simple Network Management Protocol (SNMP). Using an SNMP client (sometimes called a *network manager*) such as Sun Net Manager or HP OpenView (not provided with this product), you can monitor certain parts of the Messaging Server. Refer to [Appendix A, “SNMP Support,”](#) for details.

27.8.8 imquotacheck for Mailbox Quota Checking

You can monitor mailbox quota usage and limits by using the `imquotacheck` utility. The `imquotacheck` utility generates a report that lists defined quotas and limits, and provides information on quota usage.

For example, the following command lists all user quota information:

```
% imquotacheck
-----
Domain red.siroe.com (diskquota = not set msgquota = not set) quota usage
-----
diskquota      size(K)    %use   msgquota      msgs    %use   user
# of domains = 1
# of users = 705
no quota       50418        no quota     4392      ajonk
no quota         5          no quota      2        andrt
no quota      355518        no quota    2500      ansri
...
...
```

The following example shows the quota usage for user sorook:

```
% imquotacheck -u sorook
-----
quota usage for user sorook
-----
diskquota      size(K)    %use   msgquota      msgs    %use   user
no quota       1487        no quota     305      sorook
```

27.8.9 Monitoring Using msprobe and watcher Functions

Messaging Server provides two processes, `watcher` and `msprobe` to monitor various system services. `watcher` watches for server crashes and restarts them as necessary. `msprobe` monitors server hangs (unresponsiveness). Specifically `msprobe` monitors the following:

- **Server Response Time.** `msprobe` connects to the enabled servers using their protocol commands and measures their response times. If the response time exceeds the alarm warning threshold, an alarm message is sent (see “[27.8.9.1 Alarm Messages](#)” on page 888 to a server, or the server response time exceeds a specified timeout period, the server is restarted. Server response times are recorded in a counter database and is logged to the default log file. `counterutil` can be used to display the server response time statistics (“[27.8.3 counterutil](#)” on page 879).

The following servers are monitored by `msprobe`: `imap`, `pop`, `http`, `cert`, `job_controller`, `smtp`, `lsmtp`, `mmp` and `ens`. When `smtp` or `lsmtp` are not responding, the dispatcher is restarted. `ens` cannot be automatically restarted.

- **Disk usage.** `msprobe` checks the disk availability and usage for every message store partition. Specifically it checks the message store `mboxlist` database directory and the MTA queue directory. If disk usage exceeds a configured threshold, an alarm message is sent. The disk sizes and usages are recorded in a counter database and is logged to the default log file. Administrators can use the `counterutil` utility (see “[27.8.3 counterutil](#)” on page 879) to display the disk usage statistics.
- **Message Store `mboxlist` Database Log File Accumulation.** Log file accumulation is an indication of an `mboxlist` database error. `msprobe` counts the number of active log files and if the number of active log files is larger than the threshold, `msprobe` logs a critical error message to the default log file to inform the admin to restart the server. If the autorestart is enabled (`local.autorestart` to `yes`), the store daemon is restarted.

`watcher` and `msprobe` are controlled by the `configutil` options shown in Table 27–5. Further information can be found in “[4.5 Automatic Restart of Failed or Unresponsive Services](#)” on [page 127](#)

TABLE 27–5 `msprobe` and `watcher` `configutil` Options

Options	Description
<code>local.autorestart</code>	Enable automatic server restart. Automatically restarts failed or hung services. Default: no
<code>local.autorestart.timeout</code>	Failure retry time-out. If a server fails more than twice in this designated amount of time, then the system stops trying to restart the server. The value (set in seconds) should be set to a period value longer than the <code>msprobe</code> interval (<code>local.schedule.msprobe</code>). Default: 600 seconds
<code>local.probe.service.timeout</code>	Timeout for a specific server before restart. <i>service</i> can be <code>imap</code> , <code>pop</code> , <code>http</code> , <code>cert</code> , <code>job_controller</code> , <code>smtp</code> , <code>lsmtp</code> , <code>mmp</code> or <code>ens</code> . Default: use <code>service.readtimeout</code>
<code>local.probe.service.warningthreshold</code>	Number of seconds of a specific server’s non-response before a warning message is logged to default log file. <i>service</i> can be <code>imap</code> , <code>pop</code> , <code>http</code> , <code>cert</code> , <code>job_controller</code> , <code>smtp</code> , <code>lsmtp</code> , <code>mmp</code> or <code>ens</code> . Default: Use <code>local.probe.warningthreshold</code>
<code>local.probe.warningthreshold</code>	Number of seconds of server non-response before a warning message is logged to default log file. Default: 5 secs

TABLE 27-5 msprobe and watcher configutil Options *(Continued)*

Options	Description
local.queuedir	MTA queue directory to check if queue size exceeded threshold defined by alarm.diskavail.msgalarmthreshold. Default: none
service.readtimeout	Period of server non-response before restarting that server. See local.schedule.msprobe. Default: 10 seconds
local.schedule.msprobe	msprobe run schedule. A crontab style schedule string (see Table 20-10) Note that by default, this is automatically set. See “ 4.6.2 Pre-defined Automatic Tasks ” on page 130. To disable: set local.schedule.msprobe.enable to NO.
local.watcher.enable	Enable watcher which monitors service failures. (IMAP, POP, HTTP, job controller, dispatcher, message store (stored), imsched, and MMP. (LMTP/SMTP servers are monitored by the dispatcher and LMTP/SMTP clients are monitored by the job_controller.) Logs error messages to the default log file for specific failures. Default: on

27.8.9.1

Alarm Messages

msprobe can issue alarms in the form of email messages to the postmaster (see “[27.6.1.2 To Monitor imapd, popd and httpd](#)” on page 876) warning of a specified condition. A sample email alarm sent when a certain threshold is exceeded is shown below:

```
Subject: ALARM: server response time in seconds of "ldap_siroe.com_389" is 10
Date: Tue, 17 Jul 2001 16:37:08 -0700 (PDT)
From: postmaster@siroe.com
To: postmaster@siroe.com

Server instance: /opt/SUNWmsgsr
Alarmsid: serverresponse
Instance: ldap_siroe_europa.com_389
Description: server response time in seconds
Current measured value (17/Jul/2001:16:37:08 -0700): 10
Lowest recorded value: 0
Highest recorded value: 10
Monitoring interval: 600 seconds
Alarm condition is when over threshold of 10
Number of times over threshold: 1
```

You can specify how often msprobe monitors disk and server performance, and under what circumstances it sends alarms. This is done by using the configutil command to set the alarm

parameters. Table 27–6 shows useful alarm parameters along with their default setting. See “configutil Parameters” in *Sun Java System Messaging Server 6.3 Administration Reference*.

TABLE 27–6 Useful Alarm Message configutil Parameters

Parameter	Description (Default in parenthesis)
alarm.msgalarmnoticehost	(localhost) Machine to which you send warning messages.
alarm.msgalarmnoticeport	(25) The SMTP port to which to connect when sending alarm message.
alarm.msgalarmnoticercpt	(Postmaster@localhost) Whom to send alarm notice.
alarm.msgalarmnoticesender	(Postmaster@localhost) Address of sender the alarm.
alarm.diskavail.msgalarmdescription	(percentage mail partition diskspace available.) Text for description field for disk availability alarm.
alarm.diskavail.msgalarmstatinterval	(3600) Interval in seconds between disk availability checks. Set to 0 to disable checking of disk usage.
alarm.diskavail.msgalarmthreshold	(10) Percentage of disk space availability below which an alarm is sent.
alarm.diskavail.msgalarmthresholddirection	(-1) Specifies whether the alarm is issued when disk space availability goes below threshold (-1) or above it (1).
alarm.diskavail.msgalarmwarninginterval	(24). Interval in hours between subsequent repetition of disk availability alarms.
alarm.serverresponse.msgalarmdescription	(server response time in seconds.) Text for description field for servers response alarm.
alarm.serverresponse.msgalarmstatinterval	(600) Interval in seconds between server response checks. Set to 0 to disable checking of server response.
alarm.serverresponse.msgalarmthreshold	(10) If server response time in seconds exceeds this value, alarm issued.
alarm.serverresponse.msgalarmthresholddirection	(1) Specifies whether alarm is issued when server response time is greater than (1) or less than (-1) the threshold.
alarm.serverresponse.msgalarmwarninginterval	(24) Interval in hours between subsequent repetition of server response alarm.

SNMP Support

The Messaging Server supports system monitoring through the Simple Network Management Protocol (SNMP). Using an SNMP client (sometimes called a *network manager*) such as Sun Net Manager or HP OpenView (not provided with this product), you can monitor certain parts of the Messaging Server. For more information on monitoring the Messaging Server refer to [Chapter 27, “Monitoring Messaging Server”](#)

This chapter describes how to enable SNMP support for the Messaging Server. It also gives an overview of the type of information provided by SNMP. Note that it does not describe how to view this information from an SNMP client. Please refer to your SNMP client documentation for details on how to use it to view SNMP-based information. This document also describes some of the data available from the Messaging Server SNMP implementation, but complete MIB details are available from [RFC 2788 \(`http://www.faqs.org/rfcs/rfc2788.html`\)](http://www.faqs.org/rfcs/rfc2788.html) and [RFC 2789 \(`http://www.faqs.org/rfcs/rfc2789.html`\)](http://www.faqs.org/rfcs/rfc2789.html).

This chapter consists of the following sections:

- “[A.1 SNMP Implementation](#)” on page 891
- “[A.2 Configuring SNMP Support for Messaging Server on Solaris 9](#)” on page 893
- “[A.3 Configuring SNMP Support for Solaris 10 OS](#)” on page 894
- “[A.4 Monitoring from an SNMP Client](#)” on page 902
- “[A.5 SNMP Information from the Messaging Server](#)” on page 903

A.1 SNMP Implementation

Messaging Server implements two standardized MIBs, the Network Services Monitoring MIB (RFC 2788) and the Mail Monitoring MIB (RFC 2789). The Network Services Monitoring MIB provides for the monitoring of network services such as POP, IMAP, HTTP, and SMTP servers. The Mail Monitoring MIB provides for the monitoring of MTAs. The Mail Monitoring MIB allows for monitoring both the active and historical state of each MTA channel. The active information focuses on currently queued messages and open network connections (for

example, counts of queued messages, source IP addresses of open network connections), while the historical information provides cumulative totals (for example, total messages processed, total inbound connections).

Note – For a complete listing of Messaging Server SNMP monitoring information, refer to RFC 2788 and RFC 2789.

SNMP is supported on platforms running Solaris and Red Hat Linux. Messaging Server on the Solaris 9 Operating System uses Solstice Enterprise Agents (SEA). Starting with the Solaris 10 Operating System, Messaging Server supports the open source Net-SNMP monitoring framework, relegating the Solaris 9 OS Solstice Enterprise Agents (SEA) technology to legacy (end of support life) status. Additionally, Net-SNMP is widely used on Linux platforms. Messaging Server will use its Net-SNMP-based SNMP subagent on Solaris 10 and later as well as Linux platforms.

With the adoption of the Net-SNMP framework, Messaging Server's SNMP subagent provides new functionality:

- Support for SNMP versions 2c and 3. This support is provided by the Net-SNMP framework. The former SNMP technology, Solstice Enterprise Agents, only provided support for SNMP version 1. Enhanced security features and access controls are the primary benefit of these two versions of SNMP.
- The subagent may be configured to run as a "standalone" SNMP agent. This provides sites with additional means of isolating their various SNMP agents running on the same system.
- Multiple "instances" of Messaging Server running on the same system may concurrently be monitored. This support is provided through either Item 2 above, or through the use of SNMP version 3 "context names". This allows for SNMP monitoring of Messaging Server in failover clusters.

Limitations of the Messaging Server SNMP support are as follows:

- Only one instance of Messaging Server per host computer can be monitored via SNMP on Solaris 9 OS.
- The SNMP support is for monitoring only. No SNMP management is supported.
- No SNMP traps are implemented. (RFC 2788 provides similar functionality without using traps.)

A.1.1

SNMP Operation in the Messaging Server

The Messaging Server SNMP process is an SNMP subagent which, upon startup, registers itself with the platform's native SNMP master agent. SNMP requests from clients go to the master agent. The master agent then forwards any requests destined for the Messaging Server to the

Messaging Server subagent process. The Messaging Server subagent process then processes the request and relays the response back to the client via the master agent. This process is shown in Figure A-1.

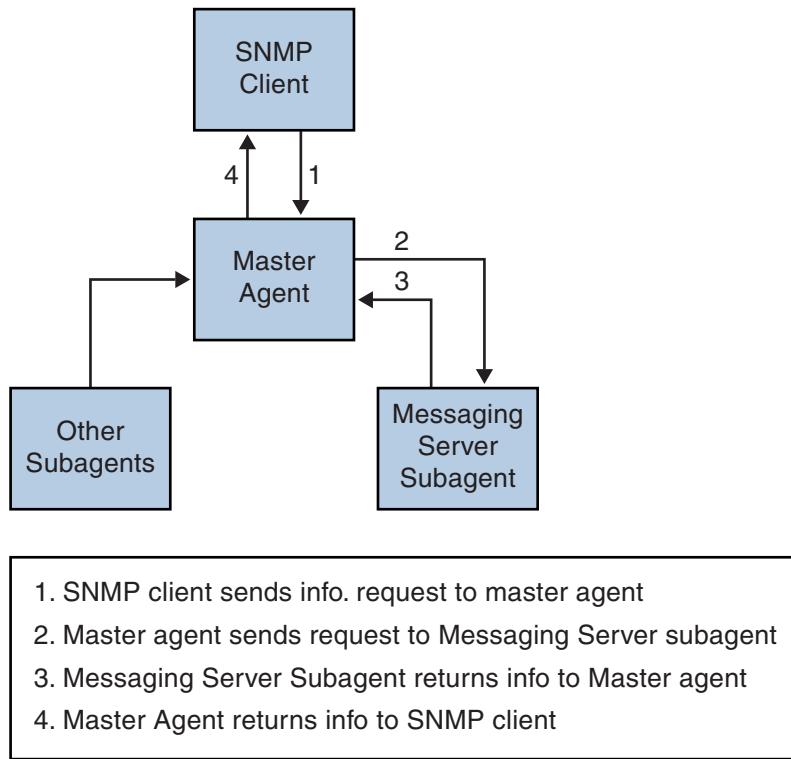


FIGURE A-1 SNMP Information Flow

A.2 Configuring SNMP Support for Messaging Server on Solaris 9

Although the overhead imposed by SNMP monitoring is very small, the Messaging Server nonetheless ships with SNMP support disabled. To enable the SNMP support, run the following commands:

```

# su user-id-for-ims
# configutil -o local.snmp.enable -v 1
# start-msg snmp

```

Once you have enabled SNMP, the `start-msg` command, without any parameters specified, will automatically start the SNMP subagent process along with the other Messaging Server processes.

Note that the Solaris native SNMP master agent must be running in order for the Messaging Server SNMP subagent to operate. The Solaris native SNMP master agent is the `snmpd` daemon which is normally started as part of the Solaris boot procedure.

The SNMP subagent will automatically select a UDP port on which to listen. Should you require, you can assign a fixed UDP port to the subagent with the following command:

```
# configutil -o local.snmp.port -v port-number
```

You may later undo this setting by specifying a value of zero for the port number. A value of zero, the default setting, tells Messaging Server to allow the subagent to automatically select any available UDP port.

Two SNMP subagent configuration files are placed in the `/etc/snmp/conf` directory: `ims.acl` which contains SNMP access control information, and `ims.reg` which contains SNMP MIB OID registration information.

Normally, there should be no reason to edit either of these files. The MIBs served out by Messaging Server are read-only and there's no need to specify a port number in the `ims.reg` file. If you do specify a port number, then it will be honored unless you also set a port number with the `configutil` utility. In that case, the port number set with `configutil` is the port number which will be used by the subagent. If you do edit the files, then you will need to stop and restart the SNMP subagent in order for your changes to take effect:

```
# stop-msg snmp  
# start-msg snmp
```

Note – Any query over SNMP made on Solaris 10 OS while SNMP support is enabled in Messaging Server, must connect to default port 16161. For example, if the open source SNMP tool `snmpwalk` is used to query network/mail statistics for Messaging Server, use the option `-p 16161`.

A.3 Configuring SNMP Support for Solaris 10 OS

By default, SNMP monitoring is disabled within Messaging Server. This default is chosen in an attempt to minimize the number of services presented by a default Messaging Server configuration. Do not interpret this default as meaning that there is a performance penalty incurred by using SNMP monitoring. Indeed, Messaging Server's SNMP support consumes very little resources and is intended to have minimal impact upon messaging server. The upshot of all of this is, of course, that one time configuration steps are required before using Messaging

Server's SNMP support. Additionally, the default configuration of the platform's Net-SNMP master agent, `snmpd`, typically needs to be changed in order to run subagents such as Messaging Servers. This change is the topic of the next section.

A.3.1

Net-SNMP Configuration

Messaging Server's Net-SNMP based SNMP subagent uses the AgentX protocol to communicate with the platform's SNMP master agent (RFC 2741). The Net-SNMP master agent, `snmpd`, must be configured to permit the use of the AgentX protocol. To do this, ensure that the platform's `snmpd.conf` file contains the line

```
master agentx
```

If that line is not present, then add it and then restart the `snmpd` daemon. Note that sending a SIGHUP signal to the daemon is not sufficient. Once the `snmpd` daemon has been restarted, look for the UNIX domain socket which `snmpd` creates for AgentX communications. On Solaris and Linux systems, this socket by default appears as the special file `/var/agentx/master`; however, its location and name may be changed via the `snmpd.conf` file.

The Solaris 10 OS `snmpd` configuration is shown below:

```
%cp /etc/sma/snmp/snmpd.conf /etc/sma/snmp/snmpd.conf.save
% cat >> /etc/sma/snmp/snmpd.conf
# Messaging Server's subagent requires the AgentX protocol
master agentx
^D
% cat >> /etc/sma/snmp/snmpd.conf
% ls -al /var/agentx/
srwxrwxrwx 1 root root 0 Aug 9 13:58 /var/agentx/master
```

Additionally, on Red Hat Enterprise Linux AS 3 systems, the default `snmpd.conf` file restricts the information which may be viewed by the "public" SNMP community. It is therefore necessary to either remove that restriction or to extend it to include the MIBs served out by Messaging Server's subagent. For initial testing, doing the latter is recommended. This is accomplished by including the OID subtrees mib-2.27 and mib-2.28 in view named "systemview" as shown in below. For actual deployment, each site must take their overall security policy into consideration. Note that the information provided by the SNMP subagent is "read only".

```
% cp /etc/snmp/snmpd.conf /etc/snmp/snmpd.conf.save
% cat >>/etc/snmp/snmpd.conf
# Messaging Server's subagent requires the AgentX protocol
master agentx
# Messaging Server's subagent exports mib-2.27 and .28
# Add the mib-2.27 and .28 OID subtrees to the systemview
```

```
view systemview included .1.3.6.1.2.1.27
view systemview included .1.3.6.1.2.1.28
^D
% /sbin/service snmpd restart
% ls -al /var/agentx/master
srwxr-xr-x 1 root root 0 Aug 8 21:20 /var/agentx/master
```

If you will be using SNMP v3 context names to distinguish between the MIBs of different instances of Messaging Server concurrently running on the same host computer, then you will also need to configure at least one SNMP v3 username and password for use with your SNMP v3 queries.

A.3.2

Messaging Server Subagent Configuration

For basic operation of Messaging Server's SNMP subagent, you need only enable it and issue a one time manual start command. Henceforth, whenever Messaging Server is started or stopped, the subagent will likewise be started or stopped. The necessary commands to effect this configuration on both Solaris and Linux are as follows:

```
% configutil -o local.snmp.enable -v 1
% start-msg snmp
```

Once running, you can test the subagent from the command line with the `snmpwalk` command. See the screen shots below an example appropriate to Solaris and Linux. Note that the files `rfc2248.txt` and `rfc2249.txt` are copies of the Network Services and MTA MIBs. On Solaris systems, these files may also be found in the `/etc/sma/snmp/mibs` directory under the names `NETWORK-SERVICES-MIB.txt` and `MTA-MIB.txt`. It is not necessary provide these files to the `snmpwalk` tool; however, doing so permits `snmpwalk` to print names for each of the MIB variables rather than their numeric object identifiers (OIDs).

Basic testing on Solaris:

```
% D=/opt/SUNWmsgsr/examples/mibs /usr/sfw/bin/snmpwalk -v 1 -c public \
-m +$D/rfc2248.txt:$D/rfc2249.txt 127.0.0.1 mib-2.27

NETWORK-SERVICES-MIB::appName.1 = STRING: /opt/SUNWmsgsr MTA on mail.siroe.com
...
% D=/opt/SUNWmsgsr/examples/mibs /usr/sfw/bin/snmpwalk -v 1 -c public \
-m +$D/rfc2248.txt:$D/rfc2249.txt 127.0.0.1 mib-2.28

MTA-MIB::mtaReceivedMessages.1 = Counter32: 1452
MTA-MIB::mtaStoredMessages.1 = Gauge32: 21
...
```

Basic testing on Linux:

```
% export D=/opt/sun/messaging/examples/mibs
% /usr/bin/snmpwalk -v 1 -c public \
    -m +$D/rfc2248.txt:$D/rfc2249.txt 127.0.0.1 mib-2.27
NETWORK-SERVICES-MIB::applName.1 = STRING: /opt/sun/messaging MTA on mail.siroe.com
...
% /usr/bin/snmpwalk -v 1 -c public \
    -m +$D/rfc2248.txt:$D/rfc2249.txt 127.0.0.1 mib-2.28
MTA-MIB::mtaReceivedMessages.1 = Counter32: 21278
MTA-MIB::mtaStoredMessages.1 = Gauge32: 7
...
```

A.3.3 Running as a Standalone SNMP Agent

Before configuring Messaging Server's SNMP subagent to run as a standalone SNMP agent, you must first decide which Ethernet interface and UDP port you wish for it to listen on for SNMP requests. By default, it will listen on all available Ethernet interfaces using UDP port 161. In most cases, you will want to change the port number so as to not interfere with the platform's SNMP master agent, `snmpd`. In some circumstances such as HA failover you will also want to change the Ethernet interface from all available interfaces -- `INADDR_ANY` -- to a specific interface identified by its IP address. These two concepts, Ethernet interface and UDP port, are controlled via the `local.snmp.listenaddr` and `local.snmp.port` options.

Once choices for the Ethernet interface and UDP port have been made, the `local.snmp.standalone` option should have its value set to one and the subagent restarted. Once restarted, it will operate as a SNMP agent independent of `snmpd` and any subagents.

For example, to run as a standalone agent listening on UDP port 9161 of the Ethernet interface with IP address 10.53.1.37, issue the commands shown below.

Configuring to run as a standalone agent:

```
% configutil -o local.snmp.port -v 9161
% configutil -o local.snmp.listenaddr -v 10.53.1.37
% configutil -o local.snmp.standalone -v 1
% stop-msg snmp
% start-msg snmp
% snmpwalk -v 1 -c public 10.53.1.37:9161 .
SNMPv2-SMI::mib-2.27.1.1.2.1 = STRING: "/opt/SUNWmsgsr MTA on mail.siroe.com"
...
```

A.3.4 Monitoring Multiple Instances of Messaging Server

Two techniques for monitoring multiple instances of Messaging Server running on the same host computer are herein discussed. The first technique, running the subagent in standalone mode, is well suited to high-availability failover (HA) configurations in which the individual

instances of Messaging Server may dynamically move between host computers. The second technique, the use of SNMP v3 context names, has some limited benefit in situations where multiple instances of Messaging Server are confined to a single system and it is desirable to limit the number of IP addresses polled by SNMP monitoring software (for example, when licensing of the monitoring software has a per IP address cost component). This latter technique may also be used in HA failover settings but would require polling just as many IP addresses as the standalone mode technique.

A.3.5 Using Standalone Agents for High-availability Failover

In a high-availability failover setting where SNMP monitoring of Messaging Server is desired, it is recommended that you run Messaging Server's SNMP subagent as a standalone agent as described in “[A.3.3 Running as a Standalone SNMP Agent](#)” on page 897. When the subagents are run in standalone mode, each HA instance of Messaging Server should have its `local.snmp.listenaddr` option set to the value of that instance's failover IP address. To simplify management, each instance should use the same UDP port, but that port should be distinct from those used by the `snmpd` daemons running on each of the physical cluster hosts. Typically those daemons will be using UDP port 161 so explicitly specify a different port number with the `local.snmp.port` option.

When Messaging Server's SNMP support is configured as recommended here, a monitoring station can monitor each instance of Messaging Server via its failover IP address or hostname regardless of which physical cluster host the instance is running on. Moreover, you are assured that Messaging Server's standalone SNMP agents will not conflict with one another as each listens only on its own virtual Ethernet interface identified by that instance's unique failover IP address. (These virtual Ethernet interfaces are automatically created by the HA failover framework.) Owing to the careful selection of a UDP port, the agents do not conflict with the `snmpd` daemons running on systems within the cluster.

A.3.6 Distinguishing Multiple Instances Through SNMP v3 Context Names

While there is no downside to using Messaging Server's SNMP support in standalone mode as described in “[A.3.3 Running as a Standalone SNMP Agent](#)” on page 897, it is recognized that some sites may prefer to use a more traditional subagent mode while still maintaining the capability of monitoring multiple instances of Messaging Server running concurrently on the same system. For instance, an SNMP monitoring system whose licensing model limits the number of IP addresses which may be polled. To achieve this goal, continue to run Messaging Server's SNMP subagent with `local.snmp.standalone` set to zero. Additionally, configure each instance of Messaging Server to use a distinct SNMP v3 context name by specifying a non-zero

value for the `local.snmp.enablecontextname` option. If a context name different than the value of `service.defaultdomain` is desired, then set the desired name with the `local.snmp.contextname` option. Once each instance of Messaging Server's SNMP subagent is restarted, they can then be monitored via SNMP v3 queries which include the proper context names. The MIBs of two instances of Messaging Server running on the same system are distinguished by the instance's SNMP v3 context name and so no MIB object identifier (OID) conflicts will arise.

A.3.7 Messaging Server's Net-SNMP-based SNMP Subagent Options

The following options apply only to Messaging Server's Net-SNMP based SNMP subagent. That subagent is used on Solaris platforms running Solaris 10 and later as well as Linux platforms. The options described below do not apply to the legacy SNMP subagent supplied for Solaris platforms running Solaris 9 and earlier operating systems.

The options described below are `configutil` options. As such, their values are inspected with a command of the form:

```
% configutil -o option-name
```

where *option-name* is the name of the option to display the value of. To set or change an option's value, use a command of the form

```
% configutil -o option-name -v option-value
```

where *option-value* is the value to be set. Changes to these options require a restart to take effect:

```
% stop-msg snmp  
% start-msg snmp
```

What follows is a description of each option along with its default value.

TABLE A-1 SNMP Subagent Options

Option (Default)	Description
------------------	-------------

TABLE A-1 SNMP Subagent Options *(Continued)*

local.snmp.enable (0)	The Messaging Server SNMP subagent will only run when this option is given a value of 1 or true in which case Messaging Server will automatically stop and start the subagent as part of its normal startup and shutdown procedures. By default this option is set to zero which disables operation of the subagent. Before enabling the subagent, ensure that the platform's master agent has been properly configured as described in “ A.3.3 Running as a Standalone SNMP Agent ” on page 897.
local.snmp.standalone (0)	Messaging Server's SNMP support normally runs as a SNMP subagent, receiving SNMP requests via the platform's SNMP master agent, snmpd. This operational mode is the default and is selected by giving this option a value of 0 or false. However, as described in “ A.3.3 Running as a Standalone SNMP Agent ” on page 897, the subagent may run in a “standalone” mode whereby it operates as a SNMP agent independent of snmpd. When run in standalone mode, the subagent--now a SNMP agent--listens directly for SNMP requests on the Ethernet interface and UDP port specified by, respectively, the local.snmp.listenaddr and local.snmp.port options. To run in this standalone mode, specify a value of 1 or TRUE for this option. Running in standalone mode does not interfere with other SNMP master or subagents running on the system.
local.snmp.listenaddr (INADDR_ANY)	Hostname or IP address of the Ethernet interface to listen for SNMP requests when running in standalone mode. By default, all available interfaces are listened on. This corresponds to specifying the value INADDR_ANY. A specific interface may be selected by specifying either the IP address or hostname associated with that interface. The interface may be either a physical interface or a virtual interface. This option is ignored when local.snmp.standalone is set to 0 or FALSE.

TABLE A-1 SNMP Subagent Options *(Continued)*

local.snmp.cachettl (30)	Time to live (TTL) in seconds for cached monitoring data. This option controls how long the subagent will report the same monitoring data before refreshing that data with new information obtained from Messaging Server. With the exception of message loop information, data is cached for no longer than 30 seconds by default. Loop information, as determined by scanning for .HELD files, is updated only once every 10 minutes. That because of the resource cost of scanning all the on-disk message queues.
local.snmp.servertimeout (5)	Note that the subagent does not continually update its monitoring data: it is only updated upon receipt of an SNMP request and the cached data has expired (that is, outlived its TTL). If the TTL is set to 30 seconds and SNMP requests are made only every five minutes, then each SNMP request will cause the subagent to obtain fresh data from Messaging Server. That is, data from Messaging Server will be obtained only once every five minutes. If, on the other hand, SNMP requests are made every 10 seconds, then the subagent will respond to some of those requests with cached data as old as 29 seconds; Messaging Server will be polled only once every 30 seconds.
local.snmp.directoryscan (1)	The subagent determines the operational status of each monitored service by actually opening TCP connections to each service and undergoing a protocol exchange. This timeout value, measured in seconds, controls how long the subagent will wait for a response to each step in the protocol exchange. By default, a timeout value of five seconds is used.

TABLE A-1 SNMP Subagent Options *(Continued)*

local.snmp.enablecontextname (0)	The subagent has the ability to register its MIBs under an SNMP v3 <i>context name</i> . When this is done, the MIBs may only be requested by a SNMP v3 client which specifies the context name in its SNMP request. Use of context names allows multiple, independent subagents to register Network Services and MTA MIBs under the same OID tree (that is, under the same SNMP master agent). See “ A.3.4 Monitoring Multiple Instances of Messaging Server ” on page 897 for further information.
local.snmp.contextname (service.defaultdomain)	To enable the use of SNMP v3 context names, specify a value of 1 or true for this option. When that is done, the subagent will default to using the value of the service.defaultdomain option for its context name. To use a different value for the context name, use the local.snmp.contextname option.

A.4 Monitoring from an SNMP Client

The base OIDs for [RFC 2788](http://www.faqs.org/rfcs/rfc2788.html) (<http://www.faqs.org/rfcs/rfc2788.html>) and [RFC 2789](http://www.faqs.org/rfcs/rfc2789.html) (<http://www.faqs.org/rfcs/rfc2789.html>) are

mib-2.27 = 1.3.6.1.2.1.27

mib-2.28 = 1.3.6.1.2.1.28

Point your SNMP client at those two OIDs and access as the “public” SNMP community.

If you wish to load copies of the MIBs into your SNMP client, ASCII copies of the MIBs are located in the *msg-svr-base/lib/config-templates* directory under the file names *rfc2788.mib* and *rfc2789.mib*. For directions on loading those MIBs into your SNMP client software, consult the SNMP client software documentation. The *SnmpAdminString* data type used in those MIBs may not be recognized by some older SNMP clients. In that case, use the equivalent files *rfc2248.mib* and *rfc2249.mib* also found in the same directory.

A.5 SNMP Information from the Messaging Server

This section summarizes the Messaging Server information provided via SNMP. It consists of the following subsection:

- “[A.5.1 applTable](#)” on page 903
- “[A.5.2 assocTable](#)” on page 905
- “[A.5.3 mtaTable](#)” on page 906
- “[A.5.4 mtaGroupTable](#)” on page 907
- “[A.5.5 mtaGroupAssociationTable](#)” on page 908
- “[A.5.6 mtaGroupErrorTable](#)” on page 909

For detailed information refer to the individual MIB tables in [RFC 2788](#) (<http://www.faqs.org/rfcs/rfc2788.html>) and [RFC 2789](#) (<http://www.faqs.org/rfcs/rfc2789.html>). Note that the RFC/MIB terminology refers to the messaging services (MTA, HTTP, etc.) as *applications* (appl), Messaging Server network connections as *associations* (assoc), and MTA channels as *MTA groups* (mtaGroups).

Note that on platforms where more than one instance of Messaging Server may be concurrently monitored, there may then be multiple sets of MTAs and servers in the `applTable`, and multiple MTAs in the other tables.

Note – The cumulative values reported in the MIBs (e.g., total messages delivered, total IMAP connections, etc.) are reset to zero after a reboot.

Each site will have different thresholds and significant monitoring values. A good SNMP client will allow you to do trend analysis and then send alerts when sudden deviations from historical trends occur.

A.5.1 `applTable`

The `applTable` provides server information. It is a one-dimensional table with one row for the MTA and an additional row for each of the following servers, if enabled: WebMail HTTP, IMAP, POP, SMTP, and SMTP Submit. This table provides version information, uptime, current operational status (up, down, congested), number of current connections, total accumulated connections, and other related data.

Below is an example of data from `applTable` (mib-2.27.1.1).

applTable:

```
applName.1 = mailsrv-1 MTA on mailsrv-1.west.sesta.com          (1)
applVersion.1 = 5.1
```

```
applUptime.1 = 7322          (2)
applOperStatus.1 = up        (3)
applLastChange.1 = 7422      (2)
applInboundAssociations.1 =  (5)
applOutboundAssociations.1 = (2)
applAccumulatedInboundAssociations.1 = 873
applAccumulatedOutboundAssociations.1 = 234
applLastInboundActivity.1 = 1054822    (2)
applLastOutboundActivity.1 = 1054222   (2)
applRejectedInboundAssociations.1 = 0   (4)
applFailedOutboundAssociations.1 = 17
applDescription.1 = Sun Java System Messaging Server 6.1
applName.2     1 = mailsrv-1 HTTP WebMail svr. mailsrv-1.sesta.com   (1)
...
applName.3 = mailsrv-1 IMAP server on mailsrv-1.west.sesta.com
...
applName.4 = mailsrv-1 POP server on mailsrv-1.west.sesta.com
...
applName.5 = mailsrv-1 SMTP server on mailsrv-1.west.sesta.com
...
applName.6 = mailsrv-1 SMTP Submit server on mailsrv-1.west.sesta.com
...
```

Notes:

1. The application (.appl*) suffixes (.1, .2, etc.) are the row numbers, applIndex. applIndex has the value 1 for the MTA, value 2 for the HTTP server, etc. Thus, in this example, the first row of the table provides data on the MTA, the second on the POP server, etc.
The name after the equal sign is the name of the Messaging Server instance being monitored. In this example, the instance name is mailsrv-1.
2. These are SNMP TimeStamp values and are the value of sysUpTime at the time of the event. sysUpTime, in turn, is the count of hundredths of seconds since the SNMP master agent was started.
3. The operational status of the HTTP, IMAP, POP, SMTP, and SMTP Submit servers is determined by actually connecting to them via their configured TCP ports and performing a simple operation using the appropriate protocol (for example, a HEAD request and response for HTTP, a HELO command and response for SMTP, and so on). From this connection attempt, the status—up (1), down (2), or congested (4)—of each server is determined.

Note that these probes appear as normal inbound connections to the servers and contribute to the value of the applAccumulatedInboundAssociations MIB variable for each server.

For the MTA, the operational status is taken to be that of the Job Controller. If the MTA is shown to be up, then the Job Controller is up. If the MTA is shown to be down, then the Job Controller is down. This MTA operational status is independent of the status of the MTA's

Service Dispatcher. The operational status for the MTA only takes on the value of up or down. Although the Job Controller does have a concept of “congested,” it is not indicated in the MTA status.

4. For the HTTP, IMAP, and POP servers the `applRejectedInboundAssociations` MIB variable indicates the number of failed login attempts and not the number of rejected inbound connection attempts.

A.5.1.1 **applTable Usage**

Monitoring server status (`applOperStatus`) for each of the listed applications is key to monitoring each server.

If it's been a long time since the MTA last inbound activity as indicated by `applLastInboundActivity`, then something may be broken preventing connections. If `applOperStatus=2` (down), then the monitored service is down. If `applOperStatus=1` (up), then the problem may be elsewhere.

A.5.2 **assocTable**

This table provides network connection information to the MTA. It is a two-dimensional table providing information about each active network connection. Connection information is not provided for other servers.

Below is an example of data from `applTable` (mib-2.27.2.1).

assocTable:

<code>assocRemoteApplication.1.1 = 129.146.198.167</code>	<code>(1)</code>
<code>assocApplicationProtocol.1.1 = applTCPProtoID.25</code>	<code>(2)</code>
<code>assocApplicationType.1.1 = peerinitiator(3)</code>	<code>(3)</code>
<code>assocDuration.1.1 = 400</code>	<code>(4)</code>

...

Notes:

In the `.x.y` suffix (1.1), `x` is the application index, `applIndex`, and indicates which application in the `applTable` is being reported on. In this case, the MTA. The `y` serves to enumerate each of the connections for the application being reported on.

1. The source IP address of the remote SMTP client.
2. This is an OID indicating the protocol being used over the network connection. `applTCPProtoID` indicates the TCP protocol. The `.n` suffix indicates the TCP port in use and `.25` indicates SMTP which is the protocol spoken over TCP port 25.

3. It is not possible to know if the remote SMTP client is a user agent (UA) or another MTA. As such, the subagent always reports `peer-initiator; ua-initiator` is never reported.
4. This is an SNMP TimeInterval and has units of hundredths of seconds. In this example, the connection has been open for 4 seconds.

A.5.2.1

assocTable Usage

This table is used to diagnose active problems. For example, if you suddenly have 200,000 inbound connections, this table can let you know where they are coming from.

A.5.3

mtaTable

This is a one-dimensional table with one row for each MTA in the `applTable`. Each row gives totals across all channels (referred to as groups) in that MTA for select variables from the `mtaGroupTable`.

Below is an example of data from `applTable` (mib-2.28.1.1).

mtaTable:

```
mtaReceivedMessages.1 = 172778
mtaStoredMessages.1 = 19
mtaTransmittedMessages.1 = 172815
mtaReceivedVolume.1 = 3817744
mtaStoredVolume.1 = 34
mtaTransmittedVolume.1 = 3791155
mtaReceivedRecipients.1 = 190055
mtaStoredRecipients.1 = 21
mtaTransmittedRecipients.1 = 3791134
mtaSuccessfulConvertedMessages.1 = 0      (1)
mtaFailedConvertedMessages.1 = 0
mtaLoopsDetected.1 = 0                  (2)
```

Notes:

The `.x` suffix (.1) provides the row number for this application in the `applTable`. In this example, .1 indicates this data is for the first application in the `applTable`. Thus, this is data on the MTA.

1. Only takes on non-zero values for the conversion channel.
2. Counts the number of .HELD message files currently stored in the MTA's message queues.

A.5.3.1

mtaTable Usage

If mtaLoopsDetected is not zero, then there is a looping mail problem. Locate and diagnose the .HELD files in the MTA queue to resolve the problem.

If the system does virus scanning with a conversion channel and rejects infected messages, then mtaSuccessfulConvertedMessages will give a count of infected messages in addition to other conversion failures.

A.5.4

mtaGroupTable

This two-dimensional table provides channel information for each MTA in the applTable. This information includes such data as counts of stored (that is, queued) and delivered mail messages. Monitoring the count of stored messages, mtaGroupStoredMessages, for each channel is critical: when the value becomes abnormally large, mail is backing up in your queues.

Below is an example of data from mtaGroupTable (mib-2.28.2.1).

mtaGroupTable:

```

mtaGroupName.1.1 = tcp_intranet           1
...
mtaGroupName.1.2 = ims-ms
...
mtaGroupName.1.3 = tcp_local
    mtaGroupDescription.1.3 = mailsrv-1 MTA tcp_local channel
    mtaGroupReceivedMessages.1.3 = 12154
    mtaGroupRejectedMessages.1.3 = 0
    mtaGroupStoredMessages.1.3 = 2
    mtaGroupTransmittedMessages.1.3 = 12148
    mtaGroupReceivedVolume.1.3 = 622135
    mtaGroupStoredVolume.1.3 = 7
    mtaGroupTransmittedVolume.1.3 = 619853
    mtaGroupReceivedRecipients.1.3 = 33087
    mtaGroupStoredRecipients.1.3 = 2
    mtaGroupTransmittedRecipients.1.3 = 32817
    mtaGroupOldestMessageStored.1.3 = 1103
    mtaGroupInboundAssociations.1.3 = 5
    mtaGroupOutboundAssociations.1.3 = 2
    mtaGroupAccumulatedInboundAssociations.1.3 = 150262
    mtaGroupAccumulatedOutboundAssociations.1.3 = 10970
    mtaGroupLastInboundActivity.1.3 = 1054822
    mtaGroupLastOutboundActivity.1.3 = 1054222
    mtaGroupRejectedInboundAssociations.1.3 = 0
    mtaGroupFailedOutboundAssociations.1.3 = 0
    mtaGroupInboundRejectionReason.1.3 =
    mtaGroupOutboundConnectFailureReason.1.3 =

```

```
mtaGroupScheduledRetry.1.3 = 0
mtaGroupMailProtocol.1.3 = applTCPProtoID.25
mtaGroupSuccessfulConvertedMessages.1.3 = 03          2
mtaGroupFailedConvertedMessages.1.3 = 0
mtaGroupCreationTime.1.3 = 0
mtaGroupHierarchy.1.3 = 0
mtaGroupOldestMessageId.1.3 = <01IFBV8AT8HYB4T6UA@red.iplanet.com>
mtaGroupLoopsDetected.1.3 = 0                      3
mtaGroupLastOutboundAssociationAttempt.1.3 = 1054222
```

Notes:

In the .x.y suffix (example: 1.1, 1.2. 1.3), x is the application index, `applIndex`, and indicates which application in the `applTable` is being reported on. In this case, the MTA. The y serves to enumerate each of the channels in the MTA. This enumeration index, `mtaGroupIndex`, is also used in the `mtaGroupAssociationTable` and `mtaGroupErrorTable` tables.

1. The name of the channel being reported on. In this case, the `tcp_intranet` channel.
2. Only takes on non-zero values for the conversion channel.
3. Counts the number of .HELD message files currently stored in this channel's message queue.

A.5.4.1

mtaGroupTable Usage

Trend analysis on *Rejected* and *Failed* might be useful in determining potential channel problems.

A sudden jump in the ratio of `mtaGroupStoredVolume` to `mtaGroupStoredMessages` could mean that a large junk mail is bouncing around the queues.

A large jump in `mtaGroupStoredMessages` could indicate unsolicited bulk email is being sent or that delivery is failing for some reason.

If the value of `mtaGroupOldestMessageStored` is greater than the value used for the undeliverable message notification times (notices channel keyword) this may indicate a message which cannot be processed even by bounce processing. Note that bounces are done nightly so you will want to use `mtaGroupOldestMessageStored > (maximum age + 24 hours)` as the test.

If `mtaGroupLoopsDetected` is greater than 0, a mail loop has been detected.

A.5.5

mtaGroupAssociationTable

This is a three-dimensional table whose entries are indices into the `assocTable`. For each MTA in the `applTable`, there is a two-dimensional sub-table. This two-dimensional sub-table has a row for each channel in the corresponding MTA. For each channel, there is an entry for each

active network connection which that channel has currently underway. The value of the entry is the index into the `assocTable` (as indexed by the entry's value and the `applIndex` index of the MTA being looked at). This indicated entry in the `assocTable` is a network connection held by the channel.

In simple terms, the `mtaGroupAssociationTable` table correlates the network connections shown in the `assocTable` with the responsible channels in the `mtaGroupTable`.

Below is an example of data from `mtaGroupAssociationTable` (mib-2.28.3.1).

mtaGroupAssociationTable:

```
mtaGroupAssociationIndex.1.3.1 = 1      1
mtaGroupAssociationIndex.1.3.2 = 2
mtaGroupAssociationIndex.1.3.3 = 3
mtaGroupAssociationIndex.1.3.4 = 4
mtaGroupAssociationIndex.1.3.5 = 5
mtaGroupAssociationIndex.1.3.6 = 6
mtaGroupAssociationIndex.1.3.7 = 7
```

Notes:

In the `.x.y.z` suffix, `x` is the application index, `applIndex`, and indicates which application in the `applTable` is being reported on. In this case, the MTA. The `y` indicates which channel of the `mtaGroupTable` is being reported on. In this example, `3` indicates the `tcp_local` channel. The `z` serves to enumerate the associations open to or from the channel.

1. The value here is an index into the `assocTable`. Specifically, `x` and this value become, respectively, the values of the `applIndex` and `assocIndex` indices into the `assocTable`. Or, put differently, this is saying that (ignoring the `applIndex`) the first row of the `assocTable` describes a network connection controlled by the `tcp_local` channel.

A.5.6

mtaGroupErrorTable

This is another three-dimensional table which gives the counts of temporary and permanent errors encountered by each channel of each MTA while attempting delivery of messages. Entries with index values of `4000000` are temporary errors while those with indices of `5000000` are permanent errors. Temporary errors result in the message being re-queued for later delivery attempts; permanent errors result in either the message being rejected or otherwise returned as undeliverable.

Below is an example of data from `mtaGroupErrorTable` (mib-2.28.5.1).

```
mtaGroupErrorTable:  
  
    mtaGroupInboundErrorCount.1.1.4000000      1 = 0  
    mtaGroupInboundErrorCount.1.1.5000000 = 0  
    mtaGroupInternalErrorCount.1.1.4000000 = 0  
    mtaGroupInternalErrorCount.1.1.5000000 = 0  
    mtaGroupOutboundErrorCount.1.1.4000000 = 0  
    mtaGroupOutboundErrorCount.1.1.5000000 = 0  
  
    mtaGroupInboundErrorCount.1.2.4000000      1 = 0  
    ...  
  
    mtaGroupInboundErrorCount.1.3.4000000      1 = 0  
    ...
```

Notes:

1. In the .x.y.z suffix, x is the application index, `applIndex`, and indicates which application in the `applTable` is being reported on. In this case, the MTA. The y indicates which channel of the `mtaGroupTable` is being reported on. In this example, 1 specifies the `tcp_intranet` channel, 2 the `ims-ms` channel, and 3 the `tcp_local` channel. Finally, the z is either 4000000 or 5000000 and indicates, respectively, counts of temporary and permanent errors encountered while attempting message deliveries for that channel.

A.5.6.1

mtaGroupErrorTable Usage

A large jump in error count may likely indicate an abnormal delivery problem. For instance, a large jump for a `tcp_channel` may indicate a DNS or network problem. A large jump for the `ims_ms` channel may indicate a delivery problem to the message store (for example, a partition is full, `stored` problem, and so on).

Administering Event Notification Service in Messaging Server

This appendix describes what you need to do to enable the Event Notification Service Publisher (ENS Publisher) and administer Event Notification Service (ENS) in Messaging Server.

This chapter/appendix contains these sections:

- “[B.1 Loading the ENS Publisher in Messaging Server](#)” on page 911
- “[B.2 Running Sample Event Notification Service Programs](#)” on page 912
- “[B.3 Administering Event Notification Service](#)” on page 913

For more information on ENS and ENS APIs, see the *Sun Java Communications Suite 5 Event Notification Service Guide*.

B.1 Loading the ENS Publisher in Messaging Server

The Event Notification Service (ENS) is the underlying publish-and-subscribe service. ENS acts as a dispatcher used by Sun Java System applications as a central point of collection for certain types of *events* that are of interest to them. Events are changes to the value of one or more properties of a resource. Any application that wants to know when these types of events occur registers with ENS, which identifies events in order and matches notifications with subscriptions.

ENS and iBiff (the ENS publisher for Messaging Server) are bundled starting with Messaging Server. By default ENS is enabled, however, iBIFF is not loaded. (See “[B.1 Loading the ENS Publisher in Messaging Server](#)” on page 911.)

In order to subscribe to notifications in Messaging Server, you need to load the `libibiff` file on the Messaging Server host then stop and restart the messaging server.

▼ To Load the ENS Publisher on Messaging Server

Perform the following steps from the command line. In these steps, the location of the Messaging Server installation directory is *msg-svr-base*, and the Messaging Server user is *inetuser*. Typical values for these variables are /opt/SUNWmsgsr, and *mailsrv*, respectively.

- 1 As *mailsrv*, run the configutil utility to load the libibiff file.

```
cd msg-svr-base  
.configutil -o "local.store.notifyplugin" -v "msg-svr-base/lib/libibiff"
```

- 2 As root, stop then restart the messaging server.

```
cd msg-svr-base/sbin  
.stop-msg  
.start-msg
```

- 3 You are now ready to receive notifications through ENS. See “[B.2 Running Sample Event Notification Service Programs](#)” on page 912

B.2 Running Sample Event Notification Service Programs

Messaging Server contains sample programs to help you learn how to receive notifications. These sample programs are located in the *msg-svr-base/examples* directory.

▼ To Run the Sample ENS Programs

- 1 Change to the *msg-svr-base/examples* directory.

- 2 Using a C compiler, compile the apub and asub examples using the *Makefile.sample* file. Set your library search path to include the *msg-svr-base/examples* directory.

- 3 Once the programs have been compiled, you can run them as follows in separate windows:

```
apub localhost 7997  
asub localhost 7997
```

Whatever is typed in the apub window should appear on the asub window. Also, if you use the default settings, all iBiff notifications should appear in the asub window.

- 4 To receive notifications published by iBiff, write a program similar to *asub.c*

For more information on the sample programs, and writing your own programs for ENS, see the *Sun Java Communications Suite 5 Event Notification Service Guide*.

Note – Once you set your library search path to include the *msg-svr-base/lib* directory, you can no longer stop and start the directory server. The workaround is to remove the entry from the library search path.

B.3 Administering Event Notification Service

Administering ENS consists of starting and stopping the service, and changing the configuration parameters to control the behavior of the iBiff publisher for ENS.

B.3.1 Starting and Stopping ENS

You use the `start-msg ens` and `stop-msg ens` commands to start and stop the ENS server. You must be root to run these commands.

- To start ENS:

msg-svr-base/sbin/start-msg ens

- To stop ENS:

msg-svr-base/sbin/stop-msg ens

▼ To Start and Stop ENS

- To start ENS:

msg-svr-base/sbin/start-msg ens

- To stop ENS:

msg-svr-base/sbin/stop-msg ens

B.3.2 Event Notification Service Configuration Parameters

Several configuration parameters control the behavior of iBiff. Use the `configutil` utility program to set these parameters.

TABLE B-1 iBiff Configuration Parameters

Parameter	Description
<code>local.store.notifyplugin.maxHeaderSize</code>	Specifies the maximum size (in bytes) of the header that will be transmitted with the notification. The default is 0 bytes.

TABLE B-1 iBiff Configuration Parameters *(Continued)*

Parameter	Description
<code>local.store.notifyplugin.maxBodySize</code>	Specifies the maximum size (in bytes) of the body that will be transmitted with the notification. The default is 0 bytes.
<code>local.store.notifyplugin.eventType.enable</code>	Specifies if the given event type will generate a notification. The legal values are 1 (to enable) and 0 (to disable). The default value is 1; that is, setting <code>local.store.notifyplugin.ReadMsg.enable</code> to 0 will disable ReadMsg notifications.
<code>local.store.notifyplugin.ensHost</code>	Specifies the hostname of the ENS server. The default is <code>127.0.0.1</code> .
<code>local.store.notifyplugin.ensPort</code>	Specifies the TCP port of the ENS server. The default is 7997.
<code>local.store.notifyplugin.ensEventKey</code>	Specifies the event key to use for ENS notifications. The default is <code>enp://127.0.0.1/store</code> . The hostname portion of the event key is not used to determine the ENS host. It is simply a unique identifier used by ENS. This key is what the subscriber should subscribe to in order to be notified of events matching this key.

Short Message Service (SMS)

This chapter describes how to implement the Short Message Service (SMS) on the SunTM ONE Messaging Server. It covers the following topics:

- “C.1 Introduction” on page 915
- “C.2 SMS Channel Theory of Operation” on page 918
- “C.3 SMS Channel Configuration” on page 934
- “C.4 SMS Gateway Server Theory of Operation” on page 963
- “C.5 SMS Gateway Server Configuration” on page 967
- “C.6 SMS Gateway Server Storage Requirements” on page 989

C.1 Introduction

Sun Java System Messaging Server implements email-to-mobile and mobile-to-email messaging using a Short Message Service (SMS). SMS can be configured to be either one-way (email-to-mobile only) or two-way (both email-to-mobile and mobile-to-email). To enable one-way service only, you must add and configure the SMS channel. To enable two-way service, you must add and configure the SMS channel, and in addition, configure the SMS Gateway Server.

For both one- and two-way SMS, the generated SMS messages are submitted to a Short Message Service Center (SMSC) using the Short Message Peer to Peer (SMPP) protocol. Specifically, the SMSC must provide a V3.4 or later SMPP server that supports TCP/IP.

Figure C-1 illustrates the logical flow of messages for both one-way and two-way SMS.

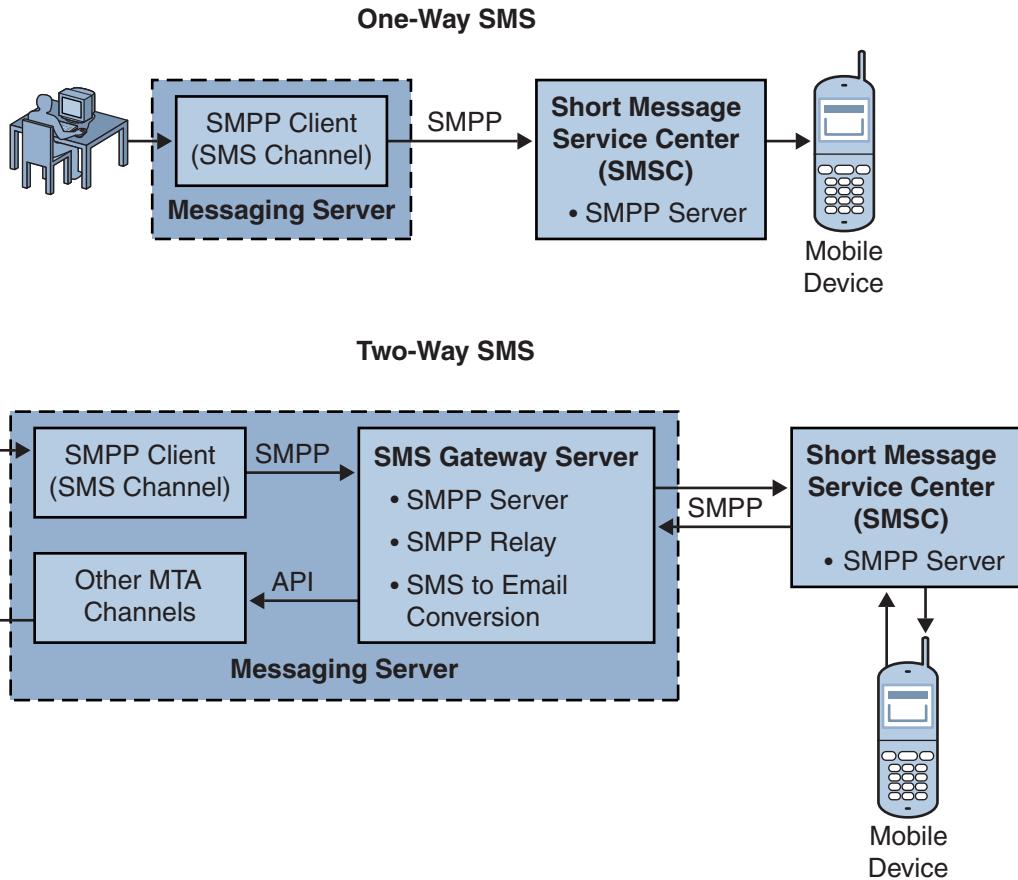


FIGURE C-1 Logical Flow For One-Way and Two-Way SMS

C.1.1 One-Way SMS

To enable one-way service, the Messaging Server implements an SMPP client (the MTA SMS channel) that communicates with remote SMSCs. The SMS channel converts enqueued email messages to SMS messages as described in “[C.2.2 The Email to SMS Conversion Process](#)” on page 919 of multipart MIME messages as well as character set translation issues.

Operating in this capacity, the SMS channel functions as an (SMPP) External Short Message Entity (ESME).

C.1.1.1 Two-Way SMS

Two-Way SMS enables the mail server not only to send email to remote devices, but allows for receiving replies from the remote devices and for remote device email origination.

Enabling two-way SMS service requires both the MTA SMS channel (SMPP client), as explained in the previous topic, and the SMS Gateway Server. Sun Java System Messaging Server installs an SMS Gateway Server as part of its general installation process, which you must then configure. The SMS Gateway Server performs two functions:

- SMPP relay

The SMS Gateway Server acts as a transparent SMPP client between the MTA SMS channel and SMSCs. However, in addition, while acting as a relay, the SMS Gateway Server generates unique SMS source addresses for relayed messages, and saves the message IDs returned by the remote SMSCs for later correlation with SMS notification messages.

- SMPP server

The SMS Gateway Server acts as an SMPP server to receive mobile originated SMS messages, replies to prior email messages, and SMS notifications. The SMS Gateway Server extracts destination email addresses from the SMS messages using profiles that define the conversion process. Profiles also describe how to handle notification messages returned by remote SMSCs in response to previously sent email-to-mobile messages.

Note – Sun Java System Messaging Server does not support the two-way SMS on the Windows platform.

C.1.2

Requirements

This manual assumes that you have read Logica CMG's SMPP specification, and the SMPP documentation for your SMSC.

In order to implement SMS, you must have the following:

- Sun Java System Messaging Server 6 or greater. (One-way SMS is also implemented in iPlanet Messaging Server 5.2.)
- The SMSC must support SMPP V3.4, or later, over TCP/IP and there must be TCP/IP connectivity between the host running Messaging Server and the SMSC.

For storage planning information for the SMS Gateway Server, see “[C.6 SMS Gateway Server Storage Requirements](#)” on page [989](#)

C.2 SMS Channel Theory of Operation

The SMS channel is a multi-threaded channel which converts queued email messages to SMS messages and then submits them for delivery to an SMSC.

The following channel operation topics are covered in this section:

- “[C.2.1 Directing Email to the Channel](#)” on page 918.
- “[C.2.2 The Email to SMS Conversion Process](#)” on page 919.
- “[C.2.3 The SMS Message Submission Process](#)” on page 925.
- “[C.2.4 Site-defined Address Validity Checks and Translations](#)” on page 929
- “[C.2.5 Site-defined Text Conversions](#)” on page 930

C.2.1 Directing Email to the Channel

When the SMS channel is configured as per “[C.3 SMS Channel Configuration](#)” on page 934 purposes of discussion, let us assume that the host name `sms.siroe.com` is a host name associated with the channel. In that case, email is directed to the channel with an address of the form:

`local-part@sms.siroe.com`

in which `local-part` is either the SMS destination address (for example, a wireless phone number, pager ID, etc.) or an attribute-value pair list in the format:

`/attribute1=value1/attribute2=value2/.../@sms.siroe.com`

The recognized attribute names and their usages are given in [Table C-1](#). These attributes allow for per-recipient control over some channel options.

TABLE C-1 SMS Attributes

Attribute Name	Attribute Value and Usage
ID	SMS destination address (for example, wireless phone number, pager ID, etc.) to direct the SMS message to. This attribute and associated value must be present.
FROM	SMS source address. Ignored when option <code>USE_HEADER_FROM=0</code> .
FROM_NPI	Use the specified NPI value. Ignored when option <code>USE_HEADER_FROM=0</code> .
FROM_TON	Use the specified TON value. Ignored when option <code>USE_HEADER_FROM=0</code> .
MAXLEN	The maximum, total bytes (that is, eight bit bytes) to place into the generated SMS message or messages for this recipient. The lower value of either MAXLEN and the value specified by the “ MAX_MESSAGE_SIZE ” on page 942 channel option is used.

TABLE C-1 SMS Attributes *(Continued)*

Attribute Name	Attribute Value and Usage
MAXPAGES	The maximum number of SMS messages to split the email message into for this recipient. The lower value of either MAXPAGES and the value specified by the “ MAX_PAGES_PER_MESSAGE ” on page 943 channel option is used.
NPI	Specify a Numeric Plan Indicator (NPI) value for the destination SMS address specified with the ID attribute. See the description of the “ DEFAULT_DESTINATION_NPI ” on page 945 channel option for information on the accepted values for this attribute. When this attribute is used, its value overrides the value given by the DEFAULT_DESTINATION_NPI channel option.
PAGELEN	Maximum number of bytes to place into a single SMS message for this recipient. The minimum of this value and that specified with the “ MAX_PAGE_SIZE ” on page 942 channel option is used.
TO	Synonym for ID.
TO_NPI	Synonym for NPI.
TO_TON	Synonym for TON.
TON	Specify a Type of Number (TON) value for the destination SMS address given with the ID attribute. See the description of the “ DEFAULT_DESTINATION_TON ” on page 946 channel option for information on the accepted values for this attribute. When this attribute is used, its value overrides the value given by the DEFAULT_DESTINATION_TON channel option.

Some example addresses:

```
123456@sms.siroe.com
/id=123456@sms.siroe.com
/id=123456/maxlen=100@sms.siroe.com
/id=123456/maxpages=1@sms.siroe.com
```

For information on performing translations, validity checks, and other operations on the SMS destination address portion of the email address, see “[C.2.4 Site-defined Address Validity Checks and Translations](#)” on page 929

C.2.2 The Email to SMS Conversion Process

In order for email to be sent to a remote site, email must be converted to SMS messages that can be understood by the remote SMSCs. This section describes the process of converting an email message queued to the SMS channel to one or more SMS messages. As described below, options allow control over the maximum number of SMS messages generated, the maximum total length of those SMS messages, and the maximum size of any single SMS message. Only text parts (that is, MIME text content types) from the email message are used and the maximum number of parts converted may also be controlled.

Character sets used in the email message’s header lines and text parts are all converted to Unicode and then converted to an appropriate SMS character set.

When there is no `SMS_TEXT` mapping table (see “[C.2.5 Site-defined Text Conversions](#)” on [page 930](#)) an email message queued to the SMS channel receives the processing illustrated in [Figure C–2](#).

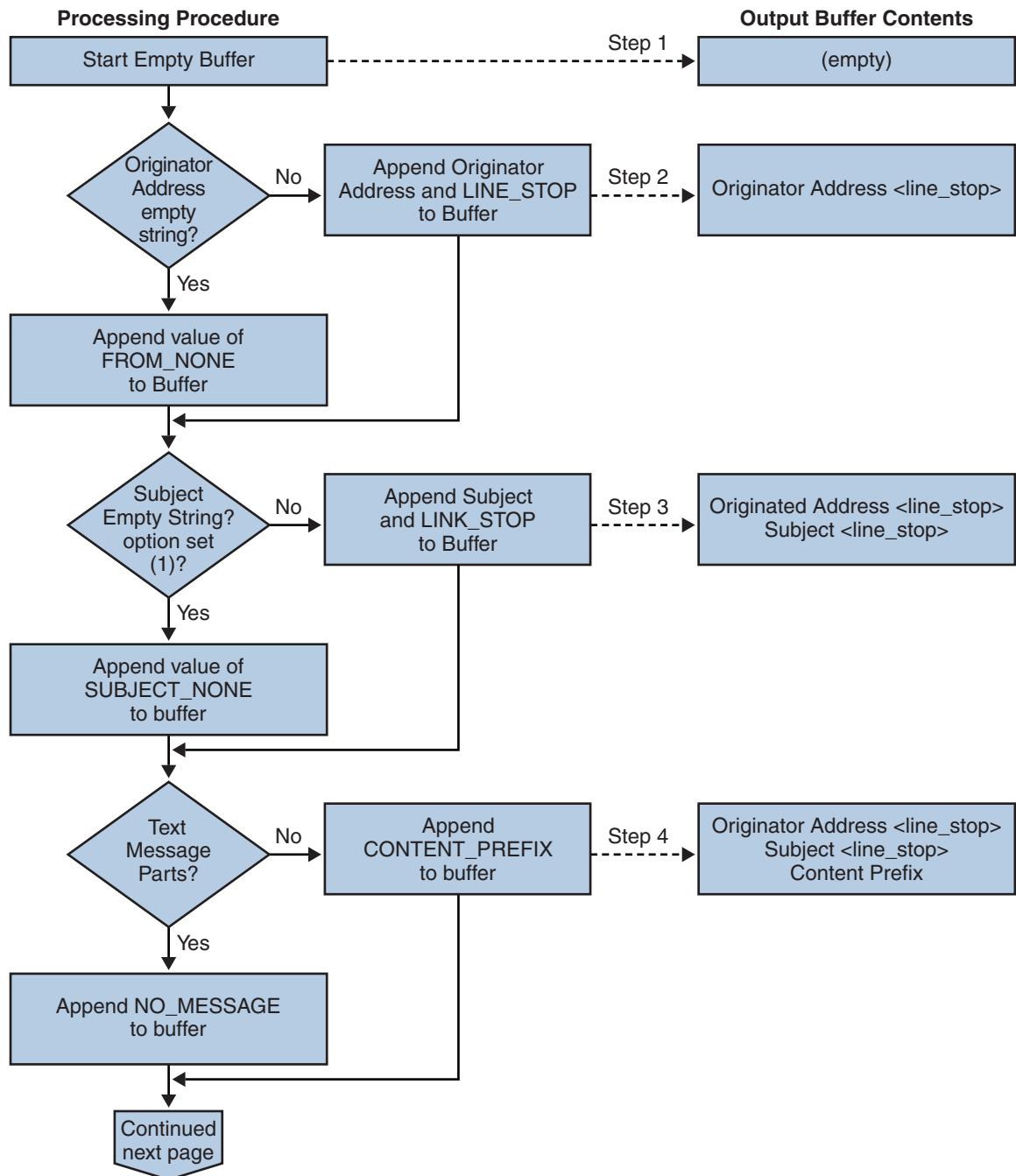
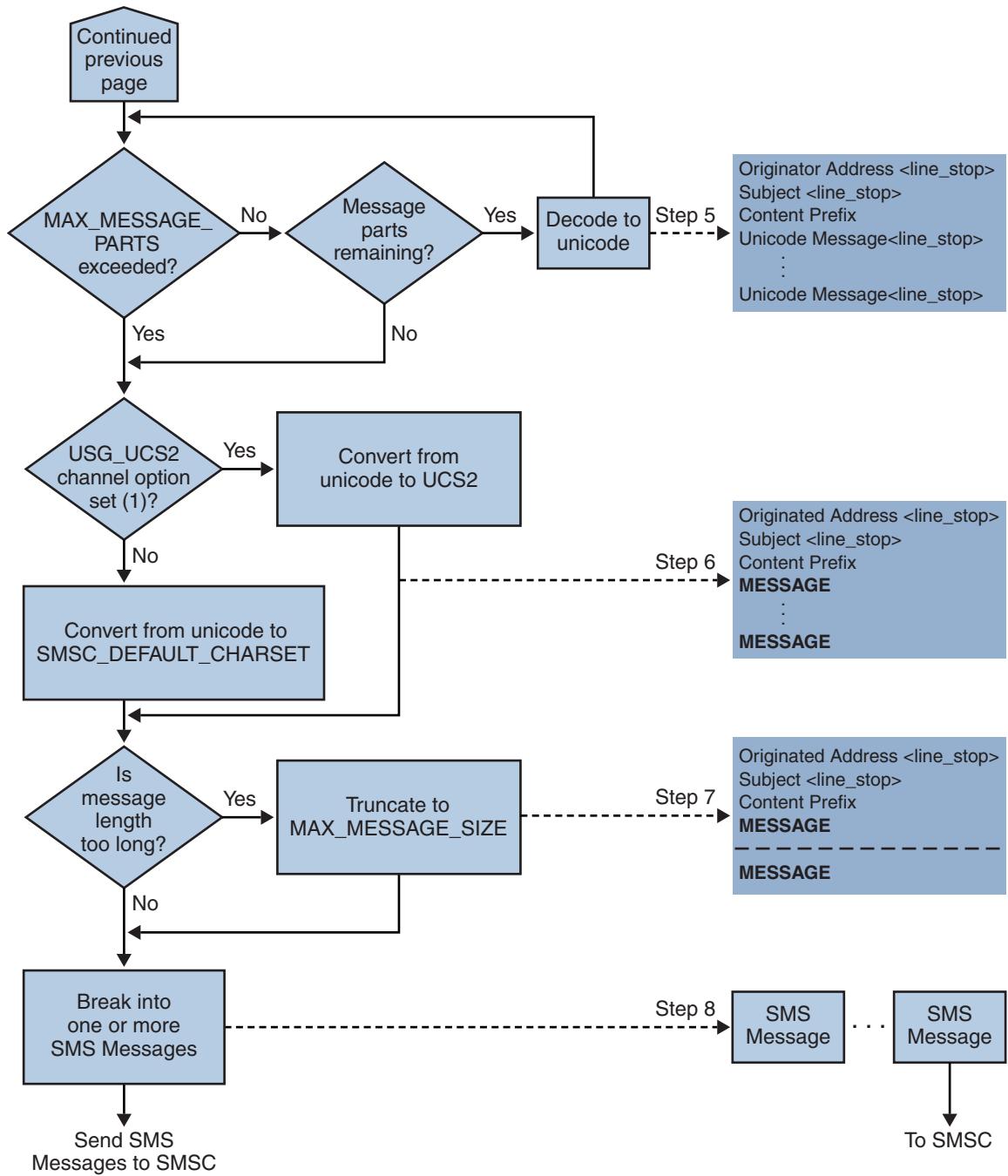


FIGURE C-2 SMS Channel Email Processing

FIGURE C-3 SMS Channel Email Processing (*continued*)

The following steps correspond to the numbered boxes in Figure C–2:

1. An empty output buffer is started. The character set used for the buffer is Unicode.
2. The email message’s originator address is taken from one of the following five sources, shown in decreasing order of preference:
 1. Resent-from:
 2. From:
 3. Resent-sender:
 4. Sender:
 5. Envelope From:

If the originator address is an empty string, then the value of the “[FROM_NONE](#)” on [page 955](#) channel option is instead appended to the buffer.

If, however, the originator address is a non-empty string, then the result of processing the “[FROM_FORMAT](#)” on [page 955](#) channel option, and the value of the [LINE_STOP](#) channel option are appended to the output buffer.

Note that the Resent-from: and Resent-sender: header lines are only considered if the “[USE_HEADER_RESENT](#)” on [page 944](#) option has the value 1. Otherwise, Resent - header lines are ignored.

3. If a Subject: header line is not present or is empty, then the value of the “[SUBJECT_NONE](#)” on [page 956](#) option is appended to the output buffer.

Otherwise, the result of processing the “[SUBJECT_FORMAT](#)” on [page 956](#) option, and the value of the “[LINE_STOP](#)” on [page 956](#) channel option are appended to the output buffer.

4. If there are no text message parts, then the value of the “[NO_MESSAGE](#)” on [page 956](#) channel option is appended to the output buffer.

If there are text message parts, then the value of the “[CONTENT_PREFIX](#)” on [page 954](#) channel option is appended to the output buffer.

Non-text message parts are discarded.

5. For each text part, while the [MAX_MESSAGE_PARTS](#) limit has not been reached, the text part is decoded to Unicode and appended to the buffer, along with the value of the [LINE_STOP](#) channel option.

6. The resulting output buffer is then converted from Unicode to either the SMSC’s default character set or UCS2 (UTF-16). The SMSC’s default character set is specified with the “[SMS_DEFAULT_CHARSET](#)” on [page 943](#) option.

7. After being converted, it is then truncated to not exceed “[MAX_MESSAGE_SIZE](#)” on [page 942](#) bytes.

8. The converted string from “[C.2.2 The Email to SMS Conversion Process](#)” on [page 919](#) is then broken into one or more SMS messages, no single SMS message longer than [MAX_PAGE_SIZE](#) bytes. At most, “[MAX_PAGES_PER_MESSAGE](#)” on [page 943](#) SMS messages will be generated.

Note – As an email message may have multiple recipients, Step 6 through Step 8 may need to be done for each recipient address which makes use of the MAXLEN, MAXPAGES, or PAGELEN attributes described in “Directing Email to the Channel,” on page 4.

C.2.2.1

Sample Email Message Processing

For example, with the channel’s default settings, the email message:

```
From: John Doe  
To: 1234567@sms.siroe.com  
Subject: Today's meeting  
Date: Fri, 26 March 2001 08:17
```

The staff meeting is at 14:30 today in the big conference room.

Would be converted to the SMS message:

```
jdoe@siroe.com (Today's meeting) The staff meeting is at 14:30 today in the big  
conference room.
```

A different set of option settings, that follows:

```
CONTENT_PREFIX=Msg:  
FROM_FORMAT=From:${pa}  
SUBJECT_FORMAT=Subj:$s
```

would instead produce:

```
From:John Doe Subj:Today's meeting Msg:The staff meeting is at 14:30 today in the  
big conference room.
```

C.2.3

The SMS Message Submission Process

Once an email message has been converted to one or more SMS messages, with possibly different sets for each recipient, the SMS messages are then submitted to the destination SMSC. The submissions are effected using SMPP V3.4 over TCP/IP. The hostname (SMPP_SERVER) of the SMPP server is taken to be the official host name associated with the SMS channel; the TCP port (SMPP_PORT) to use is specified with the port channel keyword.

When there are messages to process, the channel is started. The channel binds to the SMPP server as a transmitter, presenting the credentials specified with the ESME_channel options described in “[C.3.3.4 SMPP Options](#)” on page 951. Table C-2 lists the fields set in a BIND_TRANSMITTER PDU (Protocol Data Unit), and gives their values:

TABLE C-2 Fields in Generated in a `BIND_TRANSMITTER` PDU

Field	Value
<code>system_id</code>	“ ESME_SYSTEM_ID ” on page 952 channel option; default value is an empty string
<code>password</code>	“ ESME_PASSWORD ” on page 952 channel option; default value is an empty string
<code>system_type</code>	“ ESME_SYSTEM_TYPE ” on page 952 channel option; default value is an empty string
<code>interface_version</code>	0x34 indicating SMPP V3.4
<code>addr_ton</code>	“ ESME_ADDRESS_TON ” on page 951; default value is 0x00 indicating an unknown TON
<code>addr_npi</code>	“ ESME_ADDRESS_NPI ” on page 951; default value is 0x00 indicating an unknown NPI
<code>addr_range</code>	“ ESME_IP_ADDRESS ” on page 952 channel option; default value is an empty string

Note that the channel is multithreaded. Depending on how much mail there is to send, the channel may have multiple dequeue thread running. (There can even be multiple channel processes running.) Each thread does a `BIND_TRANSMITTER` and then on that TCP/IP connection, sends all of the SMS messages it has to send, and then sends an `UNBIND`, and then closes the connection. No attempt is made to hold a connection open for a period of idle time for potential reuse. If the remote SMPP server sends back a throttle error, then an `UNBIND` is issued, the TCP/IP connection is closed, and a new connection and `BIND` established. It behaves similarly if the remote SMPP server sends an `UNBIND` before it is finished sending its SMS messages.

The SMS messages are then submitted using SMPP `SUBMIT_SM` PDUs. If a permanent error is returned (for example, `ESME_RINVDSTADR`), then the email message is returned as undeliverable. If a temporary error is returned, then the email message is re-enqueued for a later delivery attempt. To clarify, a permanent error is one for which the condition is likely to exist indefinitely and for which repeated delivery attempts will have no positive effect, such as invalid SMS destination addresses. Whereas, a temporary error is one for which the condition is likely to not exist in the near future, such as a server down or server congested condition.

If the `USE_HEADER_FROM` option has the value 1, then the source address for the submitted SMS message is set. The value used is derived from the originating email message and is chosen to be the most likely (email) address to which any replies should be directed. Accordingly, the source address taken from one of the following seven sources, shown in decreasing order of preference:

1. `Resent-reply-to`:
2. `Resent-from`:

3. Reply-to:
4. From:
5. Resent-sender:
6. Sender:
7. Envelope From:

Note that the Resent-reply-to: and Reply-to: header lines are only considered if the “[USE_HEADER_REPLY_TO](#)” on page 944 option has the value 1. Moreover, the Resent-reply-to:, Resent-from:, and Resent-sender: header lines are only considered if the “[USE_HEADER_RESENT](#)” on page 944 option has the value 1. (Note that this means that both of those options must have the value 1 for the Resent-reply-to: header line to be considered.) The default value for both of these options is the value 0. As such, only items 4, 6, and 7 are considered by the default configuration. Finally, since the source address in an SMS message is limited to 20 bytes, the source address chosen will be truncated if it exceeds that limit.

[Table C-3](#) shows the mandatory fields set in a SUBMIT_SM PDU:

TABLE C-3 Mandatory Fields in Generated SUBMIT_SM PDUs

Field	Value
service_type	“ DEFAULT_SERVICE_TYPE ” on page 949 channel option; default value is an empty string.
source_addr_ton	“ DEFAULT_SOURCE_TON ” on page 950 channel option; if USE_HEADER_FROM=1, then this field is usually forced to the value 0x05 indicating an alphanumeric TON; otherwise, the default value is 0x01 indicating an international TON.
source_addr_npi	“ DEFAULT_SOURCE_NPI ” on page 949 channel option; default value is 0x00.
source_addr	“ DEFAULT_SOURCE_ADDRESS ” on page 949 channel option if USE_HEADER_FROM=0; otherwise, an alphanumeric string representing the originator of the email message.
dest_addr_ton	TON addressing attribute or “ DEFAULT_DESTINATION_TON ” on page 946 channel option; default value is 0x01 indicating an international TON.
dest_addr_npi	NPI addressing attribute or “ DEFAULT_SOURCE_NPI ” on page 949 channel option; default value is 0x00 indicating an unknown NPI.
dest_addr	Destination SMS address derived from the local part of the email envelope To: address; see “ C.2.1 Directing Email to the Channel ” on page 918.
esm_class	For one-way SMS, set to 0x03, indicating store and forward mode, default SMSC message type, and do not set reply path. For a two-way MSM message, set to 0x83.

TABLE C-3 Mandatory Fields in Generated **SUBMIT_SM** PDUs *(Continued)*

Field	Value
<code>protocol_id</code>	0x00; unused for CDMA and TDMA; for GSM, 0x00 indicates no Internet, but SME-to-SME protocol.
<code>priority_flag</code>	0x00 for GSM & CDMA and 0x01 for TDMA, all indicating normal priority; See the description of the “ DEFAULT_PRIORITY ” on page 947 channel option.
<code>schedule_delivery_time</code>	Empty string indicating immediate delivery.
<code>validity_period</code>	“ DEFAULT_VALIDITY_PERIOD ” on page 950 channel option; default value is an empty string indicating that the SMSC’s default should be used.
<code>registered_delivery</code>	0x00 indicating no registered delivery.
<code>replace_if_present_flag</code>	0x00 indicating that any previous SMS messages should not be replaced.
<code>data_coding</code>	0x00 for the SMSC’s default character set; 0x08 for the UCS2 character set.
<code>sm_default_msg_id</code>	0x00 indicating not to use a pre-defined message.
<code>sm_length</code>	Length and content of the SMS message; see “ C.2.2 The Email to SMS Conversion Process ” on page 919
<code>short_message</code>	Length and content of the SMS message; see “ C.2.2 The Email to SMS Conversion Process ” on page 919

Table C-4 shows the optional fields in a **SUBMIT_SM** PDU:

TABLE C-4 Optional Fields in Generated **SUBMIT_SM** PDUs

Field	Value
<code>privacy</code>	See the description of the “ DEFAULT_PRIVACY ” on page 948 channel keyword; default is to not provide this field unless the email message has a <code>Sensitivity</code> : header line
<code>sar_refnum</code>	See the description of the “ USE_SAR ” on page 951 channel keyword; default is to not provide these fields
<code>sar_total</code>	See <code>sar_refnum</code> above.
<code>sar_seqnum</code>	See <code>sar_refnum</code> above.

The channel remains bound to the SMPP server until either it has no more SMS messages to submit (the message queue is empty), or “[MAX_PAGES_PER_BIND](#)” on page 952 has been exceeded. In the latter case, a new connection is made and bind operation performed if there remain further SMS messages to send.

Note that the SMS channel is multithreaded. Each processing thread in the channel maintains its own TCP connection with the SMPP server. For example, if there are three processing threads each with SMS messages to submit, then the channel will have three open TCP connections to the SMPP server. Each connection will bind to the SMPP server as a transmitter. Moreover, any given processing thread will only have one outstanding SMS submission at a time. That is, a given thread will submit an SMS message, then wait for the submission response (that is, SUBMIT_SM_RESP PDU) before submitting another SMS message.

C.2.4

Site-defined Address Validity Checks and Translations

Sites may wish to apply validity checks or translations to SMS destination addresses encoded in the recipient email addresses described in “[C.2.1 Directing Email to the Channel](#)” on page 918

- Strip non-numeric characters (for example, translating 800.555.1212 to 8005551212)
- Prepend a prefix (for example, translating 8005551212 to +18005551212)
- Validate for correctness (for example, 123 is too short)

The first two tasks can be done specifically with the “[DESTINATION_ADDRESS_NUMERIC](#)” on page 950 and “[DESTINATION_ADDRESS_PREFIX](#)” on page 951 channel options. In general, all three of these tasks, and others can be implemented using mapping tables: either mapping table callouts in the rewrite rules or by means of a FORWARD mapping table. Using a mapping table callout in the rewrite rules will afford the most flexibility, including the ability to reject the address with a site-defined error response. The remainder of this section will focus on just such an approach -- using a mapping table callout from the rewrite rules.

Let us suppose that destination addresses need to be numeric only, be 10 or 11 digits long, and be prefixed with the string “+1”. This can be accomplished with the following rewrite rule

```
sms.siroe.com      ${X-REWRITE-SMS-ADDRESS,$U}@sms.siroe.com
sms.siroe.com      $?Invalid SMS address
```

The first rewrite rule above calls out to the site-define mapping table named X-REWRITE-SMS-ADDRESS. That mapping table is passed the local part of the email address for inspection. If the mapping process decides that the local part is acceptable, then the address is accepted and rewritten to the SMS channel. If the mapping process does not accept the local part, then the next rewrite rule is applied. Since it is a \$? rewrite rule, the address is rejected with the error text “Invalid SMS address”.

The X-REWRITE-SMS-ADDRESS mapping table is shown below. It performs the necessary validation steps for local parts in either attribute-value pair list format or just a raw SMS destination address.

X-VALIDATE-SMS-ADDRESS

```
! Iteratively strip any non-numeric characters
$_*[$ -/:--]%^* $0$2$R
! Accept the address if it is of the form 1nnnnnnnnn or nnnnnnnnnn
! In accepting it, ensure that we output +1nnnnnnnnn
  1%%%%%%%%%%%%% +1$0$1$2$3$4$5$6$7$8$9$Y
  %%%%%%%%%%%%%% +1$0$1$2$3$4$5$6$7$8$9$Y
! We didn't accept it and consequently it's invalid
  *             $N
```

X-REWRITE-SMS-ADDRESS

```
*/id=$_*/*      $C$0/id=$|X-VALIDATE-SMS-ADDRESS;$1|/$2$Y$E
*/id=$_*/*      $N
*
$C$|X-VALIDATE-SMS-ADDRESS;$0|$Y$E
*
```

With the above set up, be sure that “[DESTINATION_ADDRESS_NUMERIC](#)” on page 950 option has the value 0 (the default). Otherwise, the “+” will be stripped from the SMS destination address.

C.2.5

Site-defined Text Conversions

Sites may customize Steps 1 - 6 described in “[C.2.2 The Email to SMS Conversion Process](#)” on [page 919](#) a mapping table in the MTA’s mapping file.

The name of the mapping table should be *SMS_Channel_TEXT* where *SMS_Channel* is the name of the SMS channel; for example, *SMS_TEXT* if the channel is named *sms* or *SMS_MWAY_TEXT* if the channel is named *sms_mway*.

Two types of entries may be made in this mapping table. However, before explaining the format of those entries, let it be made clear that an understanding of how to use the mapping file is essential in order to understand how to construct and use these entries. An example mapping table is given after the description of these two types of entries.

Now, the two types of entries are:

- “[C.2.5.1 Message Header Entries](#)” on page 930
- “[C.2.5.2 Message Body Entries](#)” on page 931

C.2.5.1

Message Header Entries

These entries specify which message header lines should be included in an SMS message and how they should be abbreviated or otherwise converted. Only if a header line is successfully mapped to a string of non-zero length by one of these entries will it be included in the SMS message being generated. Each entry has the format

H|*pattern replacement-text*

If a message header line matches the pattern then it will be replaced with the replacement text replacement-text using the mapping file's pattern matching and string substitution facilities. The final result of mapping the header line will then be included in the SMS message provided that the metacharacter \$Y was specified in the replacement text. If a header line does not match any pattern string, if it maps to a string of length zero, or if the \$Y metacharacter is not specified in the replacement text, then the header line will be omitted from the SMS message. The two entries

```
H|From:* F:$0$Y
H|Subject:* S:$0$Y
```

cause the From: and Subject: header lines to be included in SMS messages with From: and Subject: abbreviated as F: and S:. The entries:

```
H|Date:* H|D:$0$R$Y
H|D:*,*%19%*:/*: H|D:$0$ $5:$6$R$Y
```

cause the Date: header line to be accepted and mapped such that, for instance, the header line

Date: Wed, 16 Dec 1992 16:13:27 -0700 (PDT)

will be converted to

D: Wed 16:13

Very complicated, iterative mappings may be built. Sites wishing to set up custom filters will first need to understand how the mapping file works. The H| in the right-hand-side of the entry may be omitted, if desired. The H| is allowed in that side so as to cut down on the number of table entries required by sets of iterative mappings.

C.2.5.2

Message Body Entries

Body mappings are not supported.

C.2.5.3

Example SMS Mapping Table

An example SMS_TEXT mapping table is shown in [Example C-1](#). The numbers inside parentheses at the end of each line correspond to the item numbers in the section titled “Explanatory Text” on page 932 that follows this table.

EXAMPLE C-1 Example SMS_TEXT Mapping Table.

SMS_TEXT

H From:*	H F:\$0\$R\$Y	(1)
----------	---------------	-----

EXAMPLE C-1 Example SMS_TEXT Mapping Table. *(Continued)*

H Subject:*	H S:\$0\$R\$Y	(1)
H F:*<*>*	H F:\$1\$R\$Y	()
H F:*(*)*	H F:\$0\$2\$R\$Y	(2)
H F:***"	H F:\$0\$2\$R\$Y	(3)
H F:@*	H F:\$0\$R\$Y	(4)
H %:\$ *	H \$0:\$1\$R\$Y	(5)
H %:*\$	H \$0:\$1\$R\$Y	(5)
H %:\$ \$ *	H \$0:\$1\$ \$2\$R\$Y	(6)
B *- -*	B \$0-\$1\$R	(7)
B *..*	B \$0.\$1\$R	(7)
B *!!*	B \$0!\$1\$R	(7)
B *??*	B \$0?\$\$1\$R	(7)
B *\$ \$ *	B \$0\$ \$1\$R	(6)
B \$ *	B \$0\$R	(5)
B *\$	B \$0\$R	(5)

Explanatory Text

The entries in the example SMS_TEXT mapping table above are explained below:

In the example above, the metacharacter \$R is used to implement and control iterative application of the mappings. By iterating on these mappings, powerful filtering is achieved. For instance, the simple mappings to remove a single leading or trailing space (6) or reduce two spaces to a single space (7) become, when taken as a whole, a filter which strips all leading and trailing spaces and reduces all consecutive multiple spaces to a single space. Such filtering helps reduce the size of each SMS message.

- These two entries cause From: and Subject: header lines to be included in an SMS message. From: and Subject: are abbreviated as, respectively, F: and S:. Some of the other entries may have further effects on From: and Subject: header lines.

This entry will reduce a From: header line containing a <...> pattern to only the text within the angle brackets. For example:

F: "John C. Doe" <jdoe@siroe.com> (Hello)

will be replaced with:

F: jdoe@siroe.com

- This entry will remove, inclusively, everything inside of a (...) pattern in a From: header line. For example:

F: "John C. Doe" <jdoe@siroe.com> (Hello)

will be replaced with:

F: "John C. Doe" <jdoe@siroe.com>

3. This entry will remove, inclusively, everything inside of a “...” pattern in a **From:** header line. For example:
F: "John C. Doe" <jdoe@siroe.com> (Hello)
 will be replaced with:
F: <jdoe@siroe.com> (Hello)
4. This entry will remove, inclusively, everything to the right of an at-sign, @, in a **From:** header line. For example:
F: "John C. Doe" <jdoe@siroe.com> (Hello)
 will be replaced with:
F: "John C. Doe" <jdoe@
5. These four entries remove leading and trailing spaces from lines in the message header and body.
6. These two entries reduce two spaces to a single space in lines of the message header and body.
7. These four entries reduce double dashes, periods, exclamation and question marks to single occurrences of the matching character. Again, this helps save bytes in an SMS message.

The order of the entries is very important. For instance, with the given ordering, the body of the message **From:** header line:

From: "John C. Doe" (Hello)

will be reduced to:

jdoe

The steps taken to arrive at this are as follows:

1. We begin with the From: header line:

From: "John C. Doe" (Hello)

The pattern in the first mapping entry matches this and produces the result:

F: "John C. Doe" (Hello)

The \$R metacharacter in the result string causes the result string to be remapped.

2. The mapping is applied to the result string of the last step. This produces:

F: jdoe@siroe.com

The \$R in the mapping causes the entire set of mappings to be re-applied to the result of this step.

3. Next, the mapping is applied producing:

F: jdoe

The \$R in the mapping causes the entire set of mappings to be re-applied to the result of this step.

4. Next, the mapping is applied producing:

F:jdoe

The \$R in the mapping causes the entire set of mappings to be re-applied to the result of this step.

5. Since no other entries match, the final result string:

F:jdoe

is incorporated into the SMS message.

Note – The `imsimta` test-mapping utility may be used to test a mapping table. For instance,

```
# imsimta test -mapping -noimage_file -mapping_file=test.txt
Enter table name: SMS_TEXT
Input string: H|From: "John C. Doe" (Hello)
Output string: H|F:jdoe
Output flags: [0,1,2,89]
Input string: ^D
#
```

For further details on the `imsimta` test utility, see “[imsimta test](#)” in *Sun Java System Messaging Server 6.3 Administration Reference*.

C.3 SMS Channel Configuration

This section gives directions on how to set up the SMS channel for both one-way (email-to-mobile) and two-way (email-to-mobile and mobile-to-email) functionality. The SMS channel is set up the same for both one-way and two-way functionality, with the exceptions noted in the topic “[C.3.7 Configuring the SMS Channel for Two-Way SMS](#)” on page 962

This section includes the following topics:

- “[C.3.1 Adding an SMS Channel](#)” on page 935
- “[C.3.2 Creating an SMS Channel Option File](#)” on page 937
- “[C.3.3 Available Options](#)” on page 938
- “[C.3.4 Adding Additional SMS Channels](#)” on page 958
- “[C.3.5 Adjusting the Frequency of Delivery Retries](#)” on page 959
- “[C.3.6 Sample One-Way Configuration \(MobileWay\)](#)” on page 960
- “[C.3.7 Configuring the SMS Channel for Two-Way SMS](#)” on page 962

C.3.1

Adding an SMS Channel

Two steps are required to add an SMS channel to a Messaging Server configuration:

1. “[C.3.1.1 Adding the Channel Definition and Rewrite Rules](#)” on page 935.
2. “[C.3.2 Creating an SMS Channel Option File](#)” on page 937.

While there are no channel options which must be set in all situations, it is likely that one or more of the following options may need to be set: “[ESME_PASSWORD](#)” on page 952, “[ESME_SYSTEM_ID](#)” on page 952, “[MAX_PAGE_SIZE](#)” on page 942, “[DEFAULT_SOURCE_TON](#)” on page 950, and “[DEFAULT_DESTINATION_TON](#)” on page 946. And, as described, the SMPP server’s hostname or IP address and TCP port must be set either through the channel definition in the `imta.cnf` file or the channel option file.

You may configure more than one SMS channel, giving different characteristics to different SMS channels. See “[C.3.4 Adding Additional SMS Channels](#)” on page 958 for further information on the use of multiple SMS channels.

Note for the instructions that follow: if you change the `imta.cnf` file you must recompile. If you change just a channel option file, there is no need to recompile.

Note also that the time before a channel change takes effect can differ depending on what the change is. Many channel option changes take effect in all channels started since the change was made, which may seem almost instantaneous since the Job Controller is often starting new channels. Some of the changes don’t take effect until you recompile and restart the SMTP server. These options are processed as a message is enqueued to the channel and not when the channel itself runs.

C.3.1.1

Adding the Channel Definition and Rewrite Rules

To add the channel definition and rewrite rules, do the following:

▼ To Add Channel Definition and Rewrite Rules

- 1 Before adding an SMS channel to the MTA’s configuration, you need to pick a name for the channel. The name of the channel may be either `sms` or `sms_x` where `x` is any case-insensitive string whose length is between one and thirty-six bytes. For example, `sms_mway`.
- 2 To add the channel definition, edit the `imta.cnf` file located in the `installation-directory/config/` directory. At the bottom of the file add a blank line followed by the two lines:

```
channel-name port p threaddepth t \
    backoff "pt2m" "pt5m" "pt10m" "pt30m" notices 1
smpp-host-name
```

where *channel-name* is the name you chose for the channel, *p* is the TCP port the SMPP server listens on, *t* is the maximum simultaneous number of SMPP server connections per delivery process, and *smpp-host-name* is the host name of the system running the SMPP server.

For example, you might specify a channel definition as follows:

```
sms_mway port 55555 threaddepth 20 \
backoff "pt2m" "pt5m" "pt10m" "pt30m" notices 1
smpp.siroe.com
```

For instructions on how to calculate *threaddepth*, see “[C.3.1.2 Controlling the Number of Simultaneous Connections](#)” on page 937

See “[C.3.5 Adjusting the Frequency of Delivery Retries](#)” on page 959 for a discussion of the *backoff* and *notices* channel keywords.

If you wish to specify an IP address rather than a host name, for *smpp-host-name*, specify a domain literal. For example, if the IP address is 127.0.0.1, then specify [127.0.0.1] for *smpp-host-name*. Alternatively, consider using the “[SMPP_SERVER](#)” on page 953 channel option.

Note – For Sun Java System Messaging Server 6.1, the use of the *master* channel keyword has been deprecated. It is ignored if present.

- 3 Once the channel definition has been added, go to the top half of the file and add a rewrite rule in this format:**

```
smpp-host-name $u@smpp-host-name
```

For example,

```
smpp.siroe.com $u@smpp.siroe.com
```

- 4 Save the `imta.cnf` file.**
- 5 Recompile the configuration with the `imsimta cnbuild` command.**
- 6 Restart the SMTP server with the `imsimta restart dispatcher` command.**
- 7 With the above configuration, email messages are directed to the channel by addressing them to `id@smpp-host-name` (for example, `123456@smpp.siroe.com`). See “[C.2.2 The Email to SMS Conversion Process](#)” on page 919 for further information on addressing.**

- 8 Optionally, if you wish to hide the SMPP server’s host name from users or associate other host names with the same channel, then add additional rewrite rules. For instance, to associate `host-name-1` and `host-name-2` with the channel, add the following to rewrite rules:**

```
host-name-1 $U%host-name-1@smpp-host-name
host-name-2 $U%host-name-2@smpp-host-name
```

For example, if the SMPP server's host name is `smpp.siroe.com` but you want users to address email to `id@sms.sesta.com`, then add the rewrite rule:

```
sms.sesta.com $U%sms.sesta.com@smpp.siroe.com
```

Note that the “[SMPP_SERVER](#)” on page 953 and “[SMPP_PORT](#)” on page 953 channel options will override the channel's official host name and port channel keyword settings. When the `SMPP_PORT` option is used, it is not necessary to also use the `port` keyword. The advantage of using these two options is that they can be put into effect and subsequently changed without needing to recompile the configuration. An additional use of the `SMPP_SERVER` option is described in “[C.3.4 Adding Additional SMS Channels](#)” on page 958.

C.3.1.2

Controlling the Number of Simultaneous Connections

The `threaddepth` channel keyword controls the number of messages to assign to each delivery thread within a delivery process. To calculate the total number of concurrent connections allowed, multiply the values of the two following options: `SMPP_MAX_CONNECTIONS` and `job_limit` ($SMPP_MAX_CONNECTIONS * job_limit$). The “[SMPP_MAX_CONNECTIONS](#)” on page 953 option controls the maximum number of delivery threads in a delivery process. And, the `job_limit` option, for the Job Controller processing pool in which the channel is run, controls the maximum number of simultaneous delivery processes.

To limit the total number of concurrent connections, you must adjust appropriately either or both of these options. For instance, if the remote SMPP server allows only a single connection, then both `SMPP_MAX_CONNECTIONS` and `job_limit` must be set to 1. When adjusting the values, it's preferable to allow `job_limit` to exceed 1.

C.3.2

Creating an SMS Channel Option File

In general, a channel option file contains site-specific parameters required for the operation of the channel. A channel option file is not required for SMS. If you determine that one is necessary for your installation, store it in a text file in the `installation-directory/config/` directory. As with other channel option files, the name of the file takes the form:

```
channel_name_option
```

For instance, if the channel is named `sms_mway` then the channel option file is:

```
installation-directory/config/sms_mway_option
```

Each option is placed on a single line in the file using the format:

```
option_name=option_value
```

For example,

```
PROFILE=GSM
SMSC_DEFAULT_CHARSET=iso-8859-1
USE_UCS2=1
```

For a list of available SMS channel options and a description of each, see “[C.3.3 Available Options](#)” on page 938

C.3.3 Available Options

The SMS channel contains a number of options which divide into six broad categories:

- *Email to SMS conversion*: Options which control the email to SMS conversion process.
- *SMS Gateway Server Option*: Gateway profile option.
- *SMS fields*: Options which control SMS-specific fields in generated SMS messages.
- *SMPP protocol*: Options associated with the use of the SMPP protocol over TCP/IP.
- *Localization*: Options which allow for localization of text fields inserted into SMS messages.
- *Miscellaneous*: Debug and logging options.

These options are summarized in the table below, and described more fully in the sections which follow.

TABLE C-5 SMS Channel Options

Email to SMS Conversion Options

Option (Page number)	Description	Default
“ GATEWAY_NOTIFICATIONS ” on page 941	Specify whether or not to convert email notification messages to SMS messages.	0
“ MAX_MESSAGE_PARTS ” on page 942	Max. number of message parts to extract from an email message	2
“ MAX_MESSAGE_SIZE ” on page 942	Maximum number of bytes to extract from an email message	960
“ MAX_PAGE_SIZE ” on page 942	Maximum number of bytes to put into a single SMS message	160
“ MAX_PAGES_PER_MESSAGE ” on page 943	Max. number of SMS messages to break an email message into	6
“ ROUTE_TO ” on page 943	Route SMS messages to the specified IP host name.	
“ SMSC_DEFAULT_CHARSET ” on page 943	The default character set used by the SMSC.	US-ASCII
“ USE_HEADER_FROM ” on page 943	Set the SMS source address	0

TABLE C-5 SMS Channel Options *(Continued)*

“USE_HEADER_PRIORITY” on page 944	Control the use of priority information from the email message’s header	1
“USE_HEADER_REPLY_TO” on page 944	Control the use of Reply-to: header lines when generating SMS source addresses	0
“USE_HEADER_RESENT” on page 944	Control the use of Resent-*: header lines when generating originator information	0
“USE_HEADER_SENSITIVITY” on page 944	Control the use of privacy information from the email message’s header	1
“USE_UCS2” on page 945	Use the UCS2 character set in SMS messages when applicable	1
<hr/>		
SMS Gateway Server Option		
“GATEWAY_PROFILE” on page 945	Match the gateway profile name configured in the SMS Gateway Server’s configuration file, sms_gateway.cnf	N/A
<hr/>		
SMS Fields Options		
“DEFAULT_DESTINATION_NPI” on page 945	Default NPI for SMS destination addresses	0x00
“DEFAULT_DESTINATION_TON” on page 946	Default TON for SMS destination addresses	0x01
“DEFAULT_PRIORITY” on page 947	Default priority setting for SMS messages	0=GSM, CDMA 1=TDMA
“DEFAULT_PRIVACY” on page 948	Default privacy value flag for SMS messages	-1
“DEFAULT_SERVICE_TYPE” on page 949	SMS application service associated with submitted SMS messages	N/A
“DEFAULT_SOURCE_ADDRESS” on page 949	Default SMS source address	0
“DEFAULT_SOURCE_NPI” on page 949	Default NPI for SMS source addresses	0x00
“DEFAULT_SOURCE_TON” on page 950	Default TON for SMS source addresses	0x01
“DEFAULT_VALIDITY_PERIOD” on page 950	Default validity period for SMS messages	N/A
“DESTINATION_ADDRESS_NUMERIC” on page 950	Reduce the destination SMS address to only the characters 0 - 9	0

TABLE C-5 SMS Channel Options (*Continued*)

“DESTINATION_ADDRESS_PREFIX” on page 951	Text string to prefix destination SMS addresses with	N/A
“PROFILE” on page 951	SMS profile to use	GSM
“USE_SAR” on page 951	Sequence multiple SMS messages using the SMS sar_ fields	0
SMPP Protocol Options		
“ESME_ADDRESS_NPI” on page 951	ESME NPI to specify when binding to the SMPP server	0x00
“ESME_ADDRESS_TON” on page 951	ESME TON to specify when binding to the SMPP server	0x00
“ESME_IP_ADDRESS” on page 952	IP address of the host running Sun Java System MessagingServer	N/A
“ESME_PASSWORD” on page 952	Password to present when binding to the SMPP server	N/A
“ESME_SYSTEM_ID” on page 952	System identification to present to the SMSC when binding	N/A
“ESME_SYSTEM_TYPE” on page 952	System type to present to the SMSC when binding	N/A
“MAX_PAGES_PER_BIND” on page 952	Maximum number of SMS messages to submit during a single session with an SMPP server	1024
“REVERSE_ORDER” on page 952	Transmission sequence of multi-part SMS messages	0
“SMPP_MAX_CONNECTIONS” on page 953	Maximum number of simultaneous SMPP server connections	20
“SMPP_PORT” on page 953	For one-way SMS, TCP port the SMPP server listens on. For two-way SMS, same TCP port used for the LISTEN_PORT for the SMPP relay.	N/A
“SMPP_SERVER” on page 953	For one-way SMS, host name of the SMPP server to connect to. For two-way SMS, set to point to the host name or IP address of the SMS Gateway server. If using the SMPP relay’s LISTEN_INTERFACE_ADDRESS option, then be sure to use the host name or IP address associated with the specified network interface address.	N/A
“TIMEOUT” on page 953	Timeout for completion of reads and writes with the SMPP server	30
Localization Options		
“CONTENT_PREFIX” on page 954	Text to introduce the content of the email message	Msg:
“DSN_DELAYED_FORMAT” on page 954	Formatting string for delivery delay notifications	an empty string
“DSN_FAILED_FORMAT” on page 954	Formatting string for delivery failure notifications	see description

TABLE C-5 SMS Channel Options *(Continued)*

“DSN_RELAYED_FORMAT” on page 955	Formatting string for relay notifications.	see description
“DSN_SUCCESS_FORMAT” on page 955	Formatting string to successful delivery notifications.	see description
“FROM_FORMAT” on page 955	Text to display indicating the originator of the email message	\$a
“FROM_NONE” on page 955	Text to display when there is no originator	N/A
“LANGUAGE” on page 955	(i-default) Language group to select text fields from	i-default
“LINE_STOP” on page 956	Text to place at the end of each line extracted from the email message	space character
“NO_MESSAGE” on page 956	Text to indicate that the message had no content	[no message]
“SUBJECT_FORMAT” on page 956	Text to display indicating the subject of the email message	\$s
“SUBJECT_NONE” on page 956	Text to display when there is no subject for the email message	N/A

Miscellaneous Options

“DEBUG” on page 956	Enable verbose debug output	6
“LISTEN_CONNECTION_MAX” on page 977	Maximum number of concurrent, inbound TCP connections to allow across all SMPP relay and server instantiations.	10,000
“LOG_PAGE_COUNT” on page 977	Controls the value recorded in the <code>mail.log</code> file's message size field to be page count instead of blocks.	0

C.3.3.1

Email to SMS Conversion Options

The following options control the conversion of email messages to SMS messages. The value range for the options are in parenthesis. In general, a given email message may be converted into one or more SMS messages. See “[C.2.2 The Email to SMS Conversion Process](#)” on page 919

GATEWAY_NOTIFICATIONS

(0 or 1) Specifies whether or not to convert email notifications to SMS notifications. Email notification messages must conform to RFCs 1892, 1893, 1894. The default value is 0.

When `GATEWAY_NOTIFICATIONS=0`, such notifications are discarded and are not converted to SMS notifications.

To enable the notifications to be converted to SMS notifications, set `GATEWAY_NOTIFICATIONS=1`. When the option set to 1, the localization options (`DSN_*_FORMAT`) control which notification types (success, failure, delay, relayed) are converted into SMS messages and sent through the gateway. (If the notification type has a value of an empty string, then that type notification is not converted into SMS messages.)

MAX_MESSAGE_PARTS

(*integer*) When converting a multi-part email message to an SMS message, only the first MAX_MESSAGE_PARTS number of text parts will be converted. The remaining parts are discarded. By default, MAX_MESSAGE_PARTS is 2. To allow an unlimited number of message parts, specify a value of -1. When a value of 0 is specified, then no message content will be placed into the SMS message. This has the effect of using only header lines from the email message (for example, Subject:) to generate the SMS message.

Note that an email message containing both text and an attachment will typically consist of two parts. Note further that only plain text message parts are converted. All other MIME content types are discarded.

MAX_MESSAGE_SIZE

(*integer*, ≥ 10) With this option, an upper limit may be placed on the total number of bytes placed into the SMS messages generated from an email message. Specifically, a maximum of MAX_MESSAGE_SIZE bytes will be used for the one or more generated SMS messages. Any additional bytes are discarded.

By default, an upper limit of 960 bytes is imposed. This corresponds to MAX_MESSAGE_SIZE=960. To allow any number of bytes, specify a value of zero.

The count of bytes used is made after converting the email message from Unicode to either the SMSC's default character set or UCS2. This means, in the case of UCS2, that a MAX_MESSAGE_SIZE of 960 bytes will yield, at most, 480 characters since each UCS2 character is at least two bytes long.

Note that the MAX_MESSAGE_SIZE and “MAX_PAGES_PER_MESSAGE” on page 943 options both serve the same purpose: to limit the overall size of the resulting SMS messages. Indeed, “MAX_PAGE_SIZE” on page 942=960 and “MAX_PAGE_SIZE” on page 942=160 implies MAX_PAGES_PER_MESSAGE=6. So why are there two different options? So as to allow control of the overall size or number of pages without having to consider the maximal size of a single SMS message, MAX_PAGE_SIZE. While this may not be important in the channel option file, it is important when using the “C.2.1 Directing Email to the Channel” on page 918 or “C.2.1 Directing Email to the Channel” on page 918 addressing attributes described in “C.2.1 Directing Email to the Channel” on page 918.

Finally, note that the smaller of the two limits of MAX_MESSAGE_SIZE and MAX_PAGE_SIZE * MAX_PAGES_PER_MESSAGE is used.

MAX_PAGE_SIZE

(*integer*, ≥ 10) The maximum number of bytes to allow in a single SMS message is controlled with the MAX_PAGE_SIZE option. By default, a value of 160 bytes is used. This corresponds to MAX_PAGE_SIZE=160.

MAX_PAGES_PER_MESSAGE

(*integer, 1 - 255*) The maximum number of SMS messages to generate for a given email message is controlled with this option. In effect, this option truncates the email message, only converting to SMS messages that part of the email message which fits into MAX_PAGES_PER_MESSAGE SMS messages. See the description of the “[MAX_PAGE_SIZE](#)” on page 942 option for further discussion.

By default, MAX_PAGES_PER_MESSAGE is set to the larger of 1 or “[MAX_MESSAGE_SIZE](#)” on page 942 divided by “[MAX_PAGE_SIZE](#)” on page 942.

ROUTE_TO

(*string, IP host name, 1-64 bytes*) All SMS messages targeted to the profile will be rerouted to the specified IP host name using an email address of the form:

SMS-destination-address@route-to

where SMS-destination-address is the SMS message’s destination address and the route-to is the IP host name specified with this option. The entire content of the SMS message is sent as the content of the resulting email message. The PARSE_RE_* options are ignored.

Note – Use of PARSE_RE_* and ROUTE_TO options are mutually exclusive. Use of both in the same gateway profile is a configuration error.

SMSC_DEFAULT_CHARSET

(*string*) With this option, the SMSC’s default character set may be specified. Use the character set names given in the file

installation-directory/config/charsets.txt

When this option is not specified, then US-ASCII is assumed. Note that the mnemonic names used in charsets.txt are defined in charnames.txt in the same directory.

When processing an email message, the header lines and text message parts are first decoded and then converted to Unicode. Next, the data is then converted to either the SMSC’s default character set or UCS2, depending on the value of the “[USE_UCS2](#)” on page 945 option and whether or not the SMS message contains at least one glyph not found in the default SMSC character set. Note that the UCS2 character set is a 16-bit encoding of Unicode and is often referred to as UTF-16.

USE_HEADER_FROM

(*integer, 0-2*) Set this option to allow the From: address to be passed to the SMSC. The value indicates where the From: address is taken from and what format it will have. [Table C-6](#) shows the allowable values and their meaning.

TABLE C-6 USE_HEADER_FROM Values

Value	Description
0	SMS source address never set from the From: address. Use attribute-value pair found
1	SMS source address set to from-local@from-domain, where the From: address is: @from-route:from-local@from-domain
2	SMS source address set to from-local, where the From: address is: @from-route:from-local@from-domain

USE_HEADER_PRIORITY

(0 or 1) This option controls handling of RFC 822 Priority: header lines. By default, information from the Priority: header line is used to set the resulting SMS message's priority flag, overriding the default SMS priority specified with the “[DEFAULT_PRIORITY](#)” on page 947 option. This case corresponds to USE_HEADER_PRIORITY=1. To disable use of the RFC 822 Priority: header line, specify USE_HEADER_PRIORITY=0.

See the description of the DEFAULT_PRIORITY option for further information on the handling the SMS priority flag.

USE_HEADER_REPLY_TO

(0 or 1) When USE_HEADER_FROM =1, this option controls whether or not a Reply-to: or Resent-reply-to: header line is considered for use as the SMS source address. By default, Reply-to: and Resent-reply-to: header lines are ignored. This corresponds to an option value of 0. To enable consideration of these header lines, use an option value of 1.

Note that RFC 2822 has deprecated the use of Reply-to: and Resent-reply-to: header lines.

USE_HEADER_RESENT

(0 or 1) When USE_HEADER_FROM =1, this option controls whether or not Resent- header lines are considered for use as the SMS source address. By default, Resent- header lines are ignored. This corresponds to an option value of 0. To enable consideration of these header lines, use an option value of 1.

Note that RFC 2822 has deprecated the use of Resent- header lines.

USE_HEADER_SENSITIVITY

(0 or 1) The USE_HEADER_SENSITIVITY option controls handling of RFC 822 Sensitivity: header lines. By default, information from the Sensitivity: header line is used to set the resulting SMS message's privacy flag, overriding the default SMS privacy specified with the “[DEFAULT_PRIVACY](#)” on page 948 option. This case, which is the default, corresponds to USE_HEADER_SENSITIVITY=1. To disable use of RFC 822 Sensitivity: header lines, specify USE_HEADER_SENSITIVITY=0.

See the description of the “[DEFAULT_PRIVACY](#)” on page 948 option for further information on the handling the SMS privacy flag.

USE_UCS2

(0 or 1) When appropriate, the channel will use the UCS2 character set in the SMS messages it generates. This is the default behavior and corresponds to USE_UCS2=1. To disable the use of the UCS2 character set, specify USE_UCS2=0. See the description of the “[SMS_DEFAULT_CHARSET](#)” on page 943 option for further information on character set issues.

TABLE C-7 Valid Values for USE_UCS2

USE_UCS2 Value	Result
1 (default)	The SMSC default character set will be used whenever possible. When the originating email message contains glyphs not in the SMSC default character set, then the UCS2 character set will be used.
0	The SMSC default character set will always be used. Glyphs not available in that character set will be represented by mnemonics (for example, “AE” for AE-ligature).

C.3.3.2 SMS Gateway Server Option

GATEWAY_PROFILE

The name of the gateway profile in the SMS Gateway Server configuration file, `sms_gateway.cnf`.

C.3.3.3 SMS Options

The following options allow for specification of SMS fields in generated SMS messages.

DEFAULT_DESTINATION_NPI

(integer, 0 - 255) By default, destination addresses will be assigned an NPI (Numeric Plan Indicator) value of zero. With this option, an alternate integer value in the range 0 to 255 may be assigned. Typical NPI values include those found in [Table C-8](#) that follows:

TABLE C-8 Numeric Plan Indicator Values

Value	Description
0	Unknown
1	ISDN (E.163, E.164)

TABLE C-8 Numeric Plan Indicator Values *(Continued)*

Value	Description
3	Data (X.121)
4	Telex (F.69)
6	Land Mobile (E.212)
8	National
9	Private
10	ERMES
14	IP address (Internet)
18	WAP client ID
>= 19	Undefined

Values for this option may be specified in one of three ways:

- A decimal value (for example, 10).
- A hexadecimal value prefixed by “0x” (for example, 0x0a).
- One of the following case-insensitive text strings (the associated decimal value is shown in parentheses): data (3), default (0), e.163 (1), e.164 (1), e.212 (6), ernes (10), f.69 (4), Internet (14), ip (14), isdn (1), land-mobile (6), national (8), private (9), telex (4), unknown (0), wap (18), x.121 (3).

DEFAULT_DESTINATION_TON

(integer, 0 - 255) By default, destination addresses will be assigned a TON (Type of Number) designator value of zero. With this option, an alternate integer value in the range 0 to 255 may be assigned. Typical TON values include those found in [Table C-9](#) that follows:

TABLE C-9 Typical TON Values

Value	Description
0	Unknown
1	International
2	National
3	Network specific
4	Subscriber number

TABLE C-9 Typical TON Values *(Continued)*

Value	Description
5	Alphanumeric
6	Abbreviated
>=7	Undefined

Values for this option may be specified in one of three ways:

- A decimal value (for example, 10)
- A hexadecimal value prefixed by “0x” (for example, 0x0a)
- One of the following case-insensitive text strings (the associated decimal value is shown in parentheses): abbreviated (6), alphanumeric (5), default (0), international (1), national (2), network-specific (3), subscriber (4), unknown (0).

DEFAULT_PRIORITY

(integer, 0 - 255) SMS messages have a mandatory priority field. The interpretation of SMS priority values is shown in [Table C-10](#) that follows:

TABLE C-10 SMS Priority Values Interpreted for Each SMS Profile Type

Value	GSM	TDMA	CDMA
0	Non-priority	Bulk	Normal
1	Priority	Normal	Interactive
2	Priority	Urgent	Urgent
3	Priority	Very urgent	Emergency

With this option, the default priority to assign to SMS messages may be specified. When not specified, a default priority of 0 is used for PROFILE=GSM and CDMA, and a priority of 1 for [“PROFILE” on page 951](#)=TDMA.

Note that if [“USE_HEADER_PRIORITY” on page 944](#)=1 and an email message has an RFC 822 Priority: header line, then the priority specified in that header line will instead be used to set the priority of the resulting SMS message. Specifically, if USE_HEADER_PRIORITY=0, then the SMS priority flag is always set in accord with the DEFAULT_PRIORITY option and the RFC 822 Priority: header line is always ignored. If USE_HEADER_PRIORITY=1, then the originating email message’s RFC 822 Priority: header line is used to set the SMS message’s priority flag. If that header line is not present, then the SMS priority flag is set using the DEFAULT_PRIORITY option.

The mapping used to translate RFC 822 Priority: header line values to SMS priority flags is shown in table that follows:

TABLE C-11 Mapping for Translating Priority Header to SMS Priority Flags

RFC 822	SMS priority flag		
Priority: value	GSM	TDMA	CDMA
Third	Non-priority (0)	Bulk (0)	Normal (0)
Second	Non-priority (0)	Bulk (0)	Normal (0)
Non-urgent	Non-priority (0)	Bulk (0)	Normal (0)
Normal	Non-priority (0)	Normal (1)	Normal (0)
Urgent	Priority (1)	Urgent (2)	Urgent (2)

DEFAULT_PRIVACY

(integer, -1, 0 - 255) Whether or not to set the privacy flag in an SMS message, and what value to use is controlled with the DEFAULT_PRIVACY and “USE_HEADER_SENSITIVITY” on page 944 options. By default, a value of -1 is used for DEFAULT_PRIVACY. Table C-12 that follows shows the result of setting the DEFAULT_PRIVACY and “USE_HEADER_SENSITIVITY” on page 944 options to various values.

TABLE C-12 Result of Values for DEFAULT_PRIVACY and USE_HEADER_SENSITIVITY

DEFAULT_PRIVACY	USE_HEADER_SENSITIVITY	Result
-1	0	The SMS privacy flag is never set in SMS messages.
n >= 0	0	The SMS privacy flag is always set to the value n. RFC 822 Sensitivity: header lines are always ignored.
-1 (default)	1 (default)	The SMS message's privacy flag is only set when the originating email message has an RFC 822 Sensitivity: header line. In that case, the SMS privacy flag is set to correspond to the Sensitivity: header line's value. This is the default.
n >= 0	1	The SMS message's privacy flag is set to correspond to the originating email message's RFC 822 Sensitivity: header line. If the email message does not have a Sensitivity: header line, then the value of the SMS privacy flag is set to n.

The SMS interpretation of privacy values is shown in Table C-13 that follows:

TABLE C-13 SMS Interpretation of Privacy Values

Value	Description
0	Unrestricted
1	Restricted
2	Confidential
3	Secret
≥ 4	Undefined

The mapping used to translate RFC 822 Sensitivity: header line values to SMS privacy values is shown in [Table C-14](#) that follows:

TABLE C-14 Mapping Translation of Sensitivity Headers to SMS Privacy Values

RFC 822 Sensitivity: value	SMS privacy value
Personal	1 (Restricted)
Private	2 (Confidential)
Company confidential	3 (Secret)

DEFAULT_SERVICE_TYPE

(*string, 0 - 5 bytes*) Service type to associate with SMS messages generated by the channel. By default, no service type is specified (that is, a zero length string). Some common service types are: CMT (cellular messaging), CPT (cellular paging), VMN (voice mail notification), VMA (voice mail alerting), WAP (wireless application protocol), and USSD (unstructured supplementary data services).

DEFAULT_SOURCE_ADDRESS

(*string, 0 - 20 bytes*) Source address to use for SMS messages generated from email messages. Note that the value specified with this option is overridden by the email message's originator address when **USE_HEADER_FROM=1**. By default, the value is disabled, that is, has a value of **0**.

DEFAULT_SOURCE_NPI

(*integer, 0 - 255*) By default, source addresses will be assigned an NPI value of zero. With this option, an alternate integer value in the range 0 to 255 may be assigned. See the description of the "[DEFAULT_DESTINATION_NPI](#)" on page [945](#) option for a table of typical NPI values.

DEFAULT_SOURCE_TON

(*integer, 0 - 255*) By default, source addresses will be assigned a TON designator value of zero. With this option, an alternate integer value in the range 0 to 255 may be assigned. See the description of the “[DEFAULT_DESTINATION_TON](#) on page 946” option for a table of typical TON values.

DEFAULT_VALIDITY_PERIOD

(*string, 0 - 252 bytes*) By default, SMS messages are not given a relative validity period; instead, they use the SMSC’s default value. Use this option to specify a different relative validity period. Values may be specified in units of seconds, minutes, hours, or days. [Table C-15](#) that follows specifies the format and description of the various values for this option:

TABLE C-15 DEFAULT_VALIDITY_PERIOD Format and Values

Format	Description
<i>nnn</i>	Implicit units of seconds; for example, 604800
<i>nnns</i>	Units of seconds; for example, 604800s
<i>NNNM</i>	Units of minutes; for example, 10080m
<i>NNNH</i>	Units of hours; for example, 168h
<i>NNND</i>	Units of days; for example, 7d

A specification of 0, 0s, 0m, 0h, or 0d may be used to select the SMSC’s default validity period. That is, when a specification of 0, 0s, 0m, 0h, or 0d is used, an empty string is specified for the validity period in generated SMS messages.

Note that this option does not accept values in UTC format.

DESTINATION_ADDRESS_NUMERIC

(0 or 1) Use this option to strip all non-numeric characters from the SMS destination address extracted from the email envelope To: address. For instance, if the envelope To: address is:

"(800) 555-1212"@sms.siroe.com

then it will be reduced to:

8005551212@sms.siroe.com

To enable this stripping, specify a value of 1 for this option. By default, this stripping is disabled which corresponds to an option value of 0. Note that when enabled, the stripping is done before any destination address prefix is added via the “[DESTINATION_ADDRESS_PREFIX](#) on page 951” option.

DESTINATION_ADDRESS_PREFIX

(*string*) In some instances, it may be necessary to ensure that all SMS destination addresses are prefixed with a fixed text string; for example, “+”. This option may be used to specify just such a prefix. The prefix will then be added to any SMS destination address which lacks the specified prefix. To prevent being stripped by the “[DESTINATION_ADDRESS_NUMERIC](#)” on page 950 option, this option is applied after the DESTINATION_ADDRESS_NUMERIC option.

PROFILE

(*string*) Specify the SMS profiling to be used with the SMSC. Possible values are GSM, TDMA, and CDMA. When not specified, GSM is assumed. This option is only used to select defaults for other channel options such as “[DEFAULT_PRIORITY](#)” on page 947 and “[DEFAULT_PRIVACY](#)” on page 948.

USE_SAR

(0 or 1) Sufficiently large email messages may need to be broken into multiple SMS messages. When this occurs, the individual SMS messages can optionally have sequencing information added using the SMS sar_fields. This produces a “segmented” SMS message which can be re-assembled into a single SMS message by the receiving terminal. Specify USE_SAR=1 to indicate that this sequencing information is to be added when applicable. The default is to not add sequencing information and corresponds to USE_SAR=0.

When USE_SAR=1 is specified, the “[REVERSE_ORDER](#)” on page 952 option is ignored.

C.3.3.4

SMPP Options

The following options allow for specification of SMPP protocol parameters. The options with names beginning with the string “ESME_” serve to identify the MTA when it acts as an External Short Message Entity (ESME); that is, when the MTA binds to an SMPP server in order to submit SMS messages to the server’s associated SMSC.

ESME_ADDRESS_NPI

(*integer, 0 - 255*) By default, bind operations will specify an ESME NPI value of zero indicating an unknown NPI. With this option, an alternate integer value in the range 0 to 255 may be assigned. See the description of the “[DEFAULT_DESTINATION_NPI](#)” on page 945 option for a table of typical NPI values.

ESME_ADDRESS_TON

(*integer, 0 - 255*) By default, bind operations will specify an ESME TON value of 0. With this option, an alternate integer value in the range 0 to 255 may be assigned. See the description of the “[DEFAULT_DESTINATION_TON](#)” on page 946 option for a table of typical TON values.

ESME_IP_ADDRESS

(string, 0 - 15 bytes) When binding to the SMPP server, the **BIND** PDU indicates that the client's (that is, ESME's) address range is an IP address. This is done by specifying a TON of 0x00 and an NPI of 0x0d. The value of the address range field is then set to be the IP address of the host running the SMS channel. Specify the IP address in dotted decimal format; for example, 127.0.0.1.

ESME_PASSWORD

(string, 0 - 8 bytes) When binding to the SMPP server, a password may be required. If so, then specify that password with this option. By default, a zero-length password string is presented.

ESME_SYSTEM_ID

(string, 0 - 15 bytes) When binding to the SMPP server, a system ID for the MTA may be supplied. By default, no system ID is specified (that is, a zero-length string is used). To specify a system ID, use this option.

ESME_SYSTEM_TYPE

(string, 0 - 12 bytes) When binding to the SMPP server, a system type for the MTA may be supplied. By default, no system type is specified (that is, a zero-length string is used).

MAX_PAGES_PER_BIND

(integer, >= 0) Some SMPP servers may limit the maximum number of SMS messages submitted during a single, bound session. In recognition of this, this option allows specification of the maximum number of SMS messages to submit during a single session. Once that limit is reached, the channel will unbind, close the TCP/IP connection, re-connect, and then rebind.

By default, a value of 1024 is used for **MAX_PAGES_PER_BIND**. Note that the channel will also detect **ESME_RTHROTTLED** errors and adjust **MAX_PAGES_PER_BIND** during a single run of the channel accordingly.

REVERSE_ORDER

(0 or 1) When an email message generates more than one SMS message, those SMS messages can be submitted to the SMSC in sequential order (**REVERSE_ORDER=0**), or reverse sequential order (**REVERSE_ORDER=1**). Reverse sequential order is useful for situations where the receiving terminal displays the last received message first. In such a case, the last received message will be the first part of the email message rather than the last. By default, **REVERSE_ORDER=1** is used.

Note that this option is ignored when “**USE_SAR**” on page 951=1 is specified.

SMPP_MAX_CONNECTIONS

(*integer, 1 - 50*) This option controls the maximum number of simultaneous SMPP connections per process. As each connection has an associated thread, this option also places a limit on the maximum number of “worker” threads per process. By default, SMPP_MAX_CONNECTIONS=20.

SMPP_PORT

(*integer, 1 - 65535*) The TCP port which the SMPP server listens on may be specified with either this option or the port channel keyword. This port number must be specified through either of these two mechanisms. If it is specified with both mechanisms, then the setting made with the SMPP_PORT option takes precedence. Note that there is no default value for this option.

For two-way SMS, make sure its the same port as the LISTEN_PORT for the SMPP relay.

SMPP_SERVER

(*string, 1 - 252 bytes*) For one-way SMS, by default, the IP host name of the SMPP server to connect to is the official host name associated with the channel; that is, the host name shown on the second line of the channel’s definition in MTA’s configuration. This option may be used to specify a different host name or IP address which will override that specified in the channel definition. When specifying an IP address, use dotted decimal notation; for example, 127.0.0.1.

For two-way SMS, set to point to the host name or IP address of the SMS Gateway Server. If using the SMPP relay’s LISTEN_INTERFACE_ADDRESS option, then be sure to use the host name or IP address associated with the specified network interface address.

TIMEOUT

(*integer, >= 2*) By default, a timeout of 30 seconds is used when waiting for data writes to the SMPP server to complete or for data to be received from the SMPP server. Use the TIMEOUT option to specify, in units of seconds, a different timeout value. The specified value should be at least 1second.

C.3.3.5

Localization Options

In constructing SMS messages, the SMS channel has a number of fixed text strings it puts into those messages. These strings, for example, introduce the email’s From: address and Subject: header line. With the channel options described in this section, versions of these strings may be specified for different languages and a default language for the channel then specified.

[Example C-2](#) shows the language part of the option file:

EXAMPLE C-2 Language Specification Part of Channel Option File

```
LANGUAGE=default-language
```

```
[language=i-default]
```

EXAMPLE C-2 Language Specification Part of Channel Option File (Continued)

```
FROM_PREFIX=From:  
SUBJECT_PREFIX=Subj:  
CONTENT_PREFIX=Msg:  
LINE_STOP= NO_MESSAGE=[no message]  
REPLY_PREFIX=Re:  
  
[language=en]  
FROM_PREFIX=From:  
SUBJECT_PREFIX=Subj:  
CONTENT_PREFIX=Msg:  
LINE_STOP=  
NO_MESSAGE=[no message]  
REPLY_PREFIX=Re:  
...
```

Within each [language=x] block, the localization options relevant to that language may be specified. If a particular option is not specified within the block, then the global value for that option is used. A localization option specified outside of a [language=x] block sets the global value for that option.

For the options listed below, the string values must be specified using either the US-ASCII or UTF-8 character sets. Note that the US-ASCII character set is a special case of the UTF-8 character set.

CONTENT_PREFIX

(*string, 0 - 252 bytes*) Text string to place in the SMS message before the content of the email message itself. Default global value is the US-ASCII string “Msg:”.

DSN_DELAYED_FORMAT

(*string, 0-256 characters*) Formatting string for delivery delay notifications. By default, an empty string is used for this option, thereby inhibiting the conversion to SMS of delay notifications. Note that “[GATEWAY_NOTIFICATIONS](#) on page 941 must be set to 1 for this option to be in effect. This option is ignored when GATEWAY_NOTIFICATIONS=0.

DSN_FAILED_FORMAT

(*string, 0-256 characters*) Formatting string for permanent delivery failure notifications. The default value of this option is the string:

```
Unable to deliver your message to $a; no further delivery attempts will be  
made.
```

To inhibit conversion of failure notifications, specify an empty string for this option. Note that “[GATEWAY_NOTIFICATIONS](#)” on page 941 must be set to 1 for this option to be in effect. This option is ignored when GATEWAY_NOTIFICATIONS=0.

DSN_RELAYED_FORMAT

(*string, 0-256 characters*) Formatting string for relay notifications. The default value is the string:

```
Your message to $a has been relayed to a messaging system which may not  
provide a final delivery confirmation
```

To inhibit conversion of relay notifications, specify an empty string for this option. Note that “[GATEWAY_NOTIFICATIONS](#)” on page 941 must be set to 1 for this option to be in effect. This option is ignored when GATEWAY_NOTIFICATIONS=0.

DSN_SUCCESS_FORMAT

(*string, 0-256 characters*) Formatting string for successful delivery notifications. The default value is the string:

```
Your message to $a has been delivered
```

To inhibit conversion of successful delivery notifications, specify an empty string for this option. Note that “[GATEWAY_NOTIFICATIONS](#)” on page 941 must be set to 1 for this option to be in effect. This option is ignored when GATEWAY_NOTIFICATIONS=0.

FROM_FORMAT

(*string, 0 - 252 bytes*) Formatting template to format the originator information to insert into the SMS message. The default global value is the US-ASCII string “\$a” which substitutes in the originator’s email address. See “[C.3.3.6 Formatting Templates](#)” on page 957

FROM_NONE

(*string, 0 - 252 bytes*) Text string to place in the SMS message when there is no originator address to display. The default global value is an empty string.

Note that normally, this option will never be used as sites will typically reject email messages which lack any originator address.

LANGUAGE

(*string, 0 - 40 bytes*) The default language group to select text strings from. If not specified, then the language will be derived from the host’s default locale specification. If the host’s locale specification is not available or corresponds to “C”, then i-default will be used. (i-default corresponds to “English text intended for an international audience.”)

LINE_STOP

(*string, 0 - 252 bytes*) Text string to place in the SMS message between lines extracted from the email message. The default global value is the US-ASCII space character, ““.

NO_MESSAGE

(*string, 0 - 252 bytes*) Text string to place in the SMS message to indicate that the email message had no content. The default global value is the US-ASCII string “[no message]“.

SUBJECT_FORMAT

(*string, 0 - 252 bytes*) Formatting template to format the content of the `Subject:` header line for display in the SMS message. The global default value for this option is the US-ASCII string “(\$s)“. See “[C.3.3.6 Formatting Templates](#)” on page [957](#) for further details.

See the `SUBJECT_NONE` option for a description of the handling when there is no `Subject:` header line or the content of that header line is an empty string.

SUBJECT_NONE

(*string, 0 - 252 bytes*) Text string to display when the originating email message either has no `Subject:` header line, or the `Subject:` header line’s value is an empty string. The default global value for this option is the empty string.

DEBUG

(*integer, bitmask*) Enable debug output. The default value is 6 which selects warning and error messages. Any non-zero value enables debug output for the channel itself, the same as specifying `master_debug` on the channel definition. [Table C-16](#) defines the bit values of the `DEBUG` bitmask.

TABLE C-16 DEBUG Bitmask

Bit	Value	Description
0-31	-1	Extremely verbose output
0	1	Informational messages
1	2	Warning messages
3	4	Error messages
3	8	Subroutine call tracing
4	16	Hash table diagnostics

TABLE C-16 DEBUG Bitmask *(Continued)*

Bit	Value	Description
5	32	I/O diagnostics, receive
6	64	I/O diagnostics, transmit
7	128	SMS to email conversion diagnostics (mobile originate and SMS notification)
8	256	PDU diagnostics, header data
9	512	PDU diagnostics, body data
10	1024	PDU diagnostics, type-length-value data
11	2048	Option processing; sends all option settings to the log file.

C.3.3.6

Formatting Templates

The formatting templates specified with the “[FROM_FORMAT](#)” on page 955, “[SUBJECT_FORMAT](#)” on page 956, and all the `DSN_*` channel options are UTF-8 strings which may contain a combination of literal text and substitution sequences. Assuming the sample email address of

`Jane Doe <user@siroe>`

The recognized substitution sequences are shown in [Table C-17](#) that follows:

TABLE C-17 Substitution Sequences

Sequence	Description
\$a	Replace with the local and domain part of the originator’s email address (for example, “ <code>user@siroe</code> ”)
\$d	Replace with the domain part of the originator’s email address (for example, “ <code>domain</code> ”)
\$p	Replace with the phrase part, if any, of the originator’s email address (for example, “ <code>Jane Doe</code> ”)
\$s	Replace with the content of the <code>Subject : </code> header line
\$u	Replace with the local part of the originator’s email address (for example, “ <code>user</code> ”)
\x	Replace with the literal character “ <code>x</code> ”

For example, the formatting template

`From: $a`

produces the text string

From: user@siroe

The construct,

`${xy:alternate text}`

may be used to substitute in the text associated with the sequence x. If that text is the empty string, the text associated with the sequence y is instead used. And, if that text is the empty string, to then substitute in the alternate text. For example, consider the formatting template

From: \${pa:unknown sender}

For the originator email address

John Doe <jdoe@siroe.com>

which has a phrase part, the template produces:

From: John Doe

However, for the address

jdoe@siroe.com

which lacks a phrase, it produces

From: jdoe@siroe.com

And for an empty originator address, it produces

From: unknown sender

C.3.4

Adding Additional SMS Channels

You may configure the MTA to have more than one SMS channel. There are two typical reasons to do this:

1. To communicate with different SMPP servers.

This is quite straightforward: just add an additional SMS channel to the configuration, being sure to (a) give it a different channel name, and (b) associate different host names with it. For example,

```
sms_mway port 55555 threadddepth 20  
smpp.siroe.com
```

```
sms_ace port 777 threadddepth 20  
sms.ace.net
```

Note that no new rewrite rule is needed. If there is no directly matching rewrite rule, Messaging Sever looks for a channel with the associated host name. For example, if the server is presented with `user@host.domain`, it would look for a channel of the name “`host.domain`”. If it finds such a channel, it routes the message there. Otherwise, it starts looking for a rewrite rule for the “`.domain`” and if none is there, then for the dot (“`.`”) rule. For more information on rewrite rules, see [Chapter 11, “Configuring Rewrite Rules.”](#)

2. To communicate with the same SMPP server but using different channel options.

To communicate with the same SMPP server, using different channel options, specify the same SMPP server in the “[SMPP_SERVER](#)” on page 953 channel option for each channel definition.

Using this mechanism is necessary since two different channels cannot have the same official host name (that is, the host name listed in the second line of the channel definition). To allow them to communicate with the same SMPP server, define two separate channels, with each specifying the same SMPP server in their `SMPP_SERVER` in their channel option files.

For example, you could have the following channel definitions,

```
sms_mway_1 port 55555 threaddepth 20
SMS-DAEMON-1
```

```
sms_mway_2 port 55555 threaddepth 20
SMS-DAEMON-2
```

and rewrite rules,

```
sms-1.siroe.com $u%sms-1.siroe.com@SMS-DAEMON-1
sms-2.siroe.com $U%sms-2.siroe.com@SMS-DAEMON-2
```

Then, to have them both use the same SMPP server, each of these two channels would specify “[SMPP_SERVER](#)” on page 953=`smp.siroe.com` in their channel option file.

C.3.5 Adjusting the Frequency of Delivery Retries

When an SMS message cannot be delivered owing to temporary errors (for example, the SMPP server is not reachable), the email message is left in the delivery queue and retried again later. Unless configured otherwise, the Job Controller will not re-attempt delivery for an hour. For SMS messaging, that is likely too long to wait. As such, it is recommended that the `backoff` channel keyword be used with the SMS channel to specify a more aggressive schedule for delivery attempts. For example,

```
sms_mway port 55555 threaddepth 20 \
    backoff "pt2m" "pt5m" "pt10m" "pt30m" notices 1
smp.siroe.com
```

With the above settings, a redelivery attempt will be made at two minutes after the first attempt. If that then fails, then five minutes after the second attempt. Then ten minutes later and finally every thirty minutes. The `notices 1` channel keyword causes the message to be returned as undeliverable if it cannot be delivered after a day.

C.3.6

Sample One-Way Configuration (MobileWay)

The MTA SMS channel may be used with any SMPP V3.4 compatible SMPP server. For purposes of illustrating an example configuration, this section explains how to configure the SMS channel for use with a MobileWay SMPP server. MobileWay (<http://www.mobileway.com/> (<http://www.mobilway.com>)) is a leading provider of global data and SMS connectivity. By routing your SMS traffic through MobileWay, you can reach SMS subscribers on most of the major SMS networks throughout the world.

When requesting an SMPP account with MobileWay, you may be asked to answer the following questions:

- IP address of your SMPP client: Supply the IP address of your Messaging Server system as seen by other domains on the Internet.
- Default validity period: This is the SMS validity period which MobileWay will use should a validity period not be specified in the SMS messages you submit. SMS messages which cannot be delivered before this validity period expires will be discarded. Supply a reasonable value (for example, 2 days, 7 days, etc.).
- Window size: This is the maximum number of SMS messages your SMPP client will submit before it will stop and wait for responses from the SMPP server before submitting any further SMS messages. You must supply a value of 1 message.
- Timezone: Specify the timezone in which your Messaging Server system operates. The timezone should be specified as an offset from GMT.
- Timeout: Not relevant to one-way SMS messaging.
- IP address and TCP port for outbind requests: Not relevant for one-way SMS messaging.

After supplying MobileWay with the answers to the above questions, they will provide you with an SMPP account and information necessary to communicate with their SMPP servers. This information includes

```
Account Address: a.b.c.d:p  
Account Login: system-id  
Account Passwd: secret
```

The Account Address field is the IP address, a.b.c.d, and TCP port number, P, of the MobileWay SMPP server you will be connecting to. Use these values for the “[SMPP_SERVER](#)” on page 953 and “[SMPP_PORT](#)” on page 953 channel options. The Account Login and Passwd

are, respectively, the values to use for the “[ESME_SYSTEM_ID](#)” on page 952 and “[ESME_PASSWORD](#)” on page 952 channel options. Using this information, your channel’s option file should include

```
SMPP_SERVER=a.b.c.d
SMPP_PORT=p
ESME_SYSTEM_ID=system-id
ESME_PASSWORD=secret
```

Now, to interoperate with MobileWay you need to make two additional option settings

```
ESME_ADDRESS_TON=0x01
DEFAULT_DESTINATION_TON=0x01
```

The rewrite rule in the `imta.cnf` file can appear as

```
sms.your-domain $u@sms.your-domain
```

And, the channel definition in the `imta.cnf` file can appear as

```
sms_mobileway
sms.your-domain
```

Once the channel option file, rewrite rule, and channel definition are in place, a test message may be sent. MobileWay requires International addressing of the form

```
+<country-code><subscriber-number>
```

For instance, to send a test message to the North American subscriber with the subscriber number (800) 555-1212, you would address your email message to

```
+18005551212@sms.your-domain
```

C.3.6.1

Debugging

To debug the channel, specify the `master_debug` channel keyword in the channel’s definition. For example,

```
sms_mway port 55555 threaddepth 20 \
backoff "pt2m" "pt5m" "pt10m" "pt30m" notices 1 master_debug
```

With the `master_debug` channel keyword, basic diagnostic information about the channel’s operation will be output to the channel’s log file. For verbose diagnostic information about the SMPP transactions undertaken by the channel, also specify

```
DEBUG=-1
```

in the channel’s option file.

C.3.7 Configuring the SMS Channel for Two-Way SMS

For general directions on configuring the SMS channel, refer to earlier topics starting with “[C.3 SMS Channel Configuration](#)” on page 934. Configure the SMS channel as though it will be talking directly to the remote SMSC, with the exceptions listed in [Table C–18](#) that follows:

TABLE C–18 Two-Way Configuration Exceptions

Exception	Explanation
master channel keyword	Remove the master channel keyword, if present. It is no longer needed for SMS channel configuration.
SMPP_SERVER	Set to point to the host name or IP address of the SMS Gateway Server. If using the SMPP relay’s LISTEN_INTERFACE_ADDRESS option (see “ C.5.7 Configuration Options ” on page 973), then be sure to use the host name or IP address associated with the specified network interface address.
SMPP_PORT	Same TCP port as used for the LISTEN_PORT setting used to instantiate the SMPP relay (see “ C.5.5.2 An SMPP Relay ” on page 970)
DEFAULT_SOURCE_ADDRESS	Pick a value and then configure the remote SMSC to route this address back to the Gateway SMPP server. In the SMS channel’s option file, specify the chosen value with this option.
GATEWAY_PROFILE	Set to match the gateway profile name. See “ C.5.5.1 A Gateway Profile ” on page 969
USE_HEADER_FROM	Set to 0.

All other channel configurations should be done as described in the SMS Channel documentation.

As mentioned in “[C.5.1 Setting Up Bidirectional SMS Routing](#)” on page 967, the remote SMSC needs to be configured to route the SMS address, defined in the DEFAULT_SOURCE_ADDRESS channel option, to the Gateway’s SMPP server using the TCP port number specified with the LISTEN_PORT option. (For how to specify the LISTEN_PORT, see “[C.5.5.3 An SMPP Server](#)” on page 971.)

Note that multiple SMS channels may use the same SMPP relay. Similarly, there need be only one SMPP server or gateway profile to handle SMS replies and notifications for multiple SMS channels. The ability to configure multiple relays, servers, and gateway profiles exists to effect different usage characteristics through configuration options.

C.4 SMS Gateway Server Theory of Operation

The SMS Gateway Server facilitates two-way SMS through mechanisms that allow mobile originated SMS messages to be matched to the correct email address. The following SMS Gateway Server topics are covered in this section:

- “[C.4.1 Function of the SMS Gateway Server](#)” on page 963
- “[C.4.2 Behavior of the SMPP Relay and Server](#)” on page 963
- “[C.4.3 Remote SMPP to Gateway SMPP Communication](#)” on page 964
- “[C.4.4 SMS Reply and Notification Handling](#)” on page 965

C.4.1

Function of the SMS Gateway Server

The SMS Gateway Server simultaneously functions as both an SMPP relay and server. It may be configured to have multiple “instantiations” of each function. For instance, it may be configured to have three different SMPP relays, each listening on different TCP ports or network interfaces and relaying to different remote SMPP servers. Similarly, it may be configured to have four different SMPP servers, each listening on different combinations of TCP ports and network interfaces.

The SMS Gateway Server may be configured with zero or more gateway profiles for sending SMS messages to email. Each gateway profile describes which destination SMS addresses match the profile, how to extract the destination email addresses from SMS messages, and various characteristics of the SMS to email conversion process. Each SMS message presented to the SMS Gateway Server through either its SMPP relay or server are compared to each profile. If a match is found, then the message is routed to email.

Finally, the gateway profiles also describe how to handle notification messages returned by remote SMSCs in response to previous email-to-mobile messages.

C.4.2

Behavior of the SMPP Relay and Server

When acting as an SMPP relay, the SMS Gateway Server attempts to be as transparent as possible, relaying all requests from local SMPP clients on to a remote SMPP server and then relaying back the remote server’s responses. There are two exceptions:

- When a local SMPP client submits a message whose SMS destination address matches one of the configured gateway profiles, the submitted SMS message is sent directly back to email; the SMS message is not relayed to a remote SMPP server.
- When a local or remote SMPP client submits a message whose SMS destination address matches a unique SMS source address previously generated by the SMPP relay, the SMS message is a reply to a previously relayed message. This reply is directed back to the originator of the original message.

Note that typically the SMS Gateway Server will be configured such that the unique SMS source addresses which it generates match one of the gateway profiles.

Note – The SMS Gateway Server’s SMPP relay is only intended for use with qualified, Sun Java System SMPP clients, that is, the Sun Java System Messaging Server’s SMS channel. It is not intended for use with arbitrary SMPP clients.

When acting as an SMPP server, the SMS Gateway Server directs SMS messages to email for three circumstances:

- The SMS messages are mobile originated and match a gateway profile.
- The SMS messages are mobile originated and the SMS destination address matches a previously generated unique SMS source address.
- The SMS messages are SMS notifications which correspond to email-to-mobile messages previously relayed by the SMS Gateway Server’s SMPP relay.

All other SMS messages are rejected by the SMPP server.

C.4.3

Remote SMPP to Gateway SMPP Communication

Remote SMPP clients communicate to the Gateway SMPP server with Protocol Data Units (PDUs). Remote SMPP clients emit request PDUs to which the Gateway SMPP server responds. The Gateway SMPP server operates synchronously. It completes the response to a request PDU before it processes the next request PDU from the connected remote SMPP client.

Table C-19 that follows lists the request PDUs the Gateway SMPP server handles, and specifies the Gateway SMPP server’s response.

TABLE C-19 SMPP Server Protocol Data Units

Request PDU	SMPP Server Response
BIND_TRANSMITTERBIND _TRANSCEIVERUNBIND	Responded to with the appropriate response PDU. Authentication credentials are ignored.
OUTBIND	Gateway SMPP server sends back a BIND_RECEIVER PDU. Authentication credentials presented are ignored.
SUBMIT_SMDATA_SM	Attempts to match the destination SMS address with either a unique SMS source address or the SELECT_RE setting of a Gateway profile. If neither is matched, the PDU is rejected with an ESME_RINVSTADR error.
DELIVER_SM	Attempts to find either the destination SMS address or the received message ID in the historical record. If neither is matched, returns the error ESME_RINVMMSGID.

TABLE C-19 SMPP Server Protocol Data Units *(Continued)*

Request PDU	SMPP Server Response
BIND_RECEIVER	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
SUBMIT_MULTI	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
REPLACE_SM	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
CANCEL_SM	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
QUERY_SM	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
QUERY_LAST_MSGS	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
QUERY_MSG_DETAILS	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
ENQUIRE_LINK	Returns ENQUIRE_LINK_RESP PDU.
ALERT_NOTIFICATION	Accepted but ignored.

C.4.4 SMS Reply and Notification Handling

The SMS Gateway Server maintains a historical record of each SMS message relayed through its SMPP relays. The need to use historical data arises from the fact that when submitting an email message to SMS it is generally not possible to convert the email address of the message's originator to an SMS source address. Since any SMS replies and notifications will be directed to this SMS source address, a problem arises. This is resolved by using automatically generated, unique SMS source addresses in relayed messages. The remote SMSCs are then configured to route these SMS source addresses back to the Gateway SMPP server.

The historical data is represented as an in-memory hash table of message IDs and generated, unique SMS source addresses. This data along with the associated email origination data are also stored on disk. The disk based storage is a series of files, each file representing HASH_FILE_ROLLOVER_PERIOD seconds of transactions (the default is 30 minutes). Each file is retained for RECORD_LIFETIME seconds (the default is 3 days). See the [Sun Java Communications Suite 5 Deployment Planning Guide](#) for a discussion of the in-memory and on-disk resource requirements of the historical data.

Each record has three components:

- Email origination data (such as, envelope From: and To: addresses). This data is supplied by the MTA SMS channel when it submits a message.
- The unique SMS source address generated by the SMPP relay and inserted into the relayed SMS message.
- The resulting received message ID returned by the remote SMSC's SMPP server when it accepts a submission.

C.4.4.1

Routing Process for SMS Replies

The Gateway SMPP relays and servers use historical records to handle SMS replies, notifications and mobile originated messages. When an SMS message is presented to the SMPP relay or server, the following routing process is followed:

1. The SMS destination address is compared against the historical record to see if there is a matching, unique SMS source address that the SMPP relay previously generated. If a match is found, see Step 6..
2. If there is no match, but the message is an SMS notification (SMPP DELIVER_SM PDU), then the received message ID, if present, is compared against the historical record. If a match is found, go to Step 8. [The SMS Gateway Server actually allows these to be presented to either the SMPP relay or SMPP server.]
3. If there is no match, then the destination SMS address is compared against the SELECT_RE option expressions for each configured gateway profile. If a match is found, then go to Step 9.
4. If there is no match and the SMS message was presented to the Gateway SMPP relay, then the message is relayed to the remote SMPP server.
5. If there is no match and the SMS message was presented to the Gateway SMPP server, then the message is determined to be an invalid message and an error response is returned in the SMPP response PDU. For email to SMS, a Non Delivery Notification (NDN) is eventually generated.
6. If a matching unique SMS source address was found, then the SMS message is further inspected to see if it is a reply or a notification message. To be a notification message it must be a SUBMIT_SM PDU with a received message ID. Otherwise, it is considered to be a reply.
7. If it is a reply then the SMS message is converted to an email message using the origination email information from the historical record.
8. If it is a notification, then the SMS message is converted to an email Delivery Status Notification (DSN) as per RFC 1892-1894. Note that the ESMTP NOTIFY flags (RFC 1891) of the original email message will be honored (For example, if the SMS message is a “success” DSN but the original email message requested only “failure” notifications, then the SMS notification will be discarded).
9. If the destination SMS address matches a SELECT_RE option in a configured gateway profile, then the SMS message is treated as a mobile originated message and converted back to email message as per the PARSE_RE_n rules for that gateway profile. If the conversion fails, then the SMS message is invalid and an error response is returned.

C.5 SMS Gateway Server Configuration

This section gives directions on how to set up the SMS Gateway Server for both email-to-mobile and mobile-to-email functionality. This section includes the following topics:

- “[C.5.1 Setting Up Bidirectional SMS Routing](#)” on page 967
- “[C.5.2 Enabling and Disabling the SMS Gateway Server](#)” on page 968
- “[C.5.3 Starting and Stopping the SMS Gateway Server](#)” on page 968
- “[C.5.4 SMS Gateway Server Configuration File](#)” on page 969
- “[C.5.5 Configuring Email-To-Mobile on the Gateway Server](#)” on page 969
- “[C.5.6 Configuring Mobile-to-Email Operation](#)” on page 971
- “[C.5.7 Configuration Options](#)” on page 973
- “[C.5.8 Global Options](#)” on page 973
- “[C.5.9 SMPP Relay Options](#)” on page 977
- “[C.5.10 SMPP Server Options](#)” on page 980
- “[C.5.11 Gateway Profile Options](#)” on page 981
- “[C.5.12 Configuration Example for Two-Way SMS](#)” on page 986

C.5.1

Setting Up Bidirectional SMS Routing

The recommended way to set up bidirectional email and SMS routing between the MTA and SMSC is a three step process:

- “[C.5.1.1 Set the SMS Address Prefix](#)” on page 967– Choose an SMS address prefix. Any prefix may be used, so long as it is ten characters or less.
- “[C.5.1.2 Set the Gateway Profile](#)” on page 968– Reserve the prefix for use with the SMS Gateway Server (by setting the gateway profile).
- “[C.5.1.3 Configure the SMSC](#)” on page 968– Configure the SMSC to route SMS destination addresses to the SMS Gateway SMPP server that start with the prefix. Mobile originated email will have only the prefix. Replies and notifications will have the prefix followed by exactly ten decimal digits.

C.5.1.1

Set the SMS Address Prefix

The source SMS addresses generated by the MTA SMS channel should be set to match the selected SMS address prefix. Do this by setting the following:

- MTA SMS channel options:

`USE_HEADER_FROM=0`

`DEFAULT_SOURCE_ADDRESS=prefix`

The first setting causes the channel to not attempt to set the SMS source address from information contained in the email message. The second setting causes the SMS source address to be set (to the selected prefix) when it is not set from any other source.

- Recognize the prefix as an SMS destination address to accept and route to email. Do this by specifying the SELECT_RE gateway profile option as follows:

```
SELECT_RE=prefix
```

C.5.1.2

Set the Gateway Profile

The SMS Gateway Server's gateway profile should then be set to make all relayed SMS source addresses unique. This is the default setting but may be explicitly set by specifying the gateway profile option MAKE_SOURCE_ADDRESSES_UNIQUE=1. This will result in relayed SMS source addresses of the form:

```
prefixnnnnnnnnnn
```

where nnnnnnnnnn will be a unique, ten digit decimal number.

C.5.1.3

Configure the SMSC

Finally, the SMSC should be configured to route all SMS destination addresses matching the prefix (either just the prefix, or the prefix plus a ten digit number) to the SMS Gateway Server's SMPP server. The regular expression for such a routing will be similar to:

```
prefix([0-9]{10,10}){0,1}
```

where prefix is the value of DEFAULT_SOURCE_ADDRESS, [0-9] specifies the allowed values for the ten digit number, {10, 10} specifies that there will be a minimum of ten digits and a maximum of ten digits, and {0, 1} specifies that there can be zero or one of the 10-digit numbers.

C.5.2

Enabling and Disabling the SMS Gateway Server

- To enable the SMS Gateway Server, the configutil parameter local.smsserver.enable must be set to the value 1. Use the following configuration utility command to set it:

```
# configutil -o local.smsserver.enable -v 1
```

- To disable the gateway server, set local.smsserver.enable to the value 0, using the following command:

```
# configutil -o local.smsserver.enable -v 0
```

C.5.3

Starting and Stopping the SMS Gateway Server

After the SMS Gateway Server is enabled, it may be started and stopped with the following commands:

```
# start-msg sms
```

and

```
# stop-msg sms
```

C.5.4 SMS Gateway Server Configuration File

In order to operate, the SMS Gateway Server requires a configuration file. The configuration file is a Unicode text file encoded using UTF-8. The file can be an ASCII text file. The name of the file must be:

```
installation-directory/config/sms_gateway.cnf
```

Each option setting in the file has the following format:

```
option-name=option-value
```

Options that are part of an option group appear in the following format:

```
[group-type=group-name]
option-name-1=option-value-1
option-name-2=option-value-2
...
option-name-n=option-value-n
```

C.5.5 Configuring Email-To-Mobile on the Gateway Server

To implement the email-to-mobile part of two-way SMS, you must configure the following:

- “C.5.5.1 A Gateway Profile” on page 969
- “C.5.5.2 An SMPP Relay” on page 970
- “C.5.5.3 An SMPP Server” on page 971

C.5.5.1 A Gateway Profile

To configure an email-to-mobile gateway profile, follow these steps:

▼ To Configure an Email-to-mobile Gateway Profile

1 Add a gateway profile to the SMS Gateway Server configuration file.

To add an option group, use the following format:

```
[GATEWAY_PROFILE=profile_name]
option-name-1=option-value-1
option-name-2=option-value-2a
...
option-name-n=option-value-n
```

The length of the gateway profile name, `profile_name` in the preceding format, must not exceed 11 bytes. The name must be the same as the name for the `GATEWAY_PROFILE` channel option in the SMS channel option file. The name is case insensitive. For a list of the valid channel options, see “[C.3.3 Available Options](#)” on page 938

- 2 **Set the gateway profile options (for example, `SMSC_DEFAULT_CHARSET`) to match characteristics of the remote SMSC.**

- 3 **Set the other gateway profile options to match the SMS channel’s email characteristics.**

A complete description of gateway profile options, see “[C.5.11 Gateway Profile Options](#)” on page 981

- 4 **Set the CHANNEL option.**

Set its value to the name of the MTA SMS channel.

When a notification is sent to email through the gateway, the resulting email message will be enqueued to the MTA using this channel name.

C.5.5.2

An SMPP Relay

To configure an SMPP Relay, complete the following steps:

▼ To Configure an SMPP Relay

- 1 **Add an SMPP relay instantiation (option group) to the SMS Gateway Server’s configuration file.**

To add an option group, use the following format:

```
[SMPP_RELAY=relay_name]
option-name-1=option-value-1
option-name-2=option-value-2
...
option-name-n=option-value-n
```

Any name may be used for the relay’s name. All that matters is that the name not be used for any other SMPP relay instantiation within the same configuration file.

- 2 **Set the LISTEN_PORT option.**

The value used for the SMS channel’s `SMPP_PORT` option must match that used for the relay’s `LISTEN_PORT` option. For the `LISTEN_PORT`, select a TCP port number which is not used by any other SMPP relay or server instantiation nor by any other server running on the same computer.

- 3 **Set the SERVER_HOST option.**

The relay’s `SERVER_HOST` option should give the host name for the remote SMSC’s SMPP server. An IP address may be used in place of a host name.

4 Set the SERVER_PORT option.

The relay's SERVER_PORT option should give the TCP port for the remote SMSC's SMPP server.

For a complete description of all SMPP relay options, see “[C.5.9 SMPP Relay Options](#)” on page [977](#)

C.5.3

An SMPP Server

To configure an SMPP server, complete the following steps:

▼ **To Configure an SMPP Server**

1 Add an SMPP server instantiation (option group) to the SMS Gateway Server's configuration file.

To add an option group, use the following format:

```
[SMPP_SERVER=server_name]
option-name-1=option-value-1
option-name-2=option-value-2...
option-name-n=option-value-n
```

Any name may be used for the server's name. All that matters is that the name not be used for any other SMPP server instantiation within the same configuration file.

2 Set LISTEN_PORT option.

Select a TCP port number which is not being used by any other server or relay instantiation. Additionally, the port number must not be being used by any other server on the same computer.

The remote SMSC needs to be configured to route notifications via SMPP to the SMS Gateway Server system using this TCP port.

For a complete description of all SMPP server options, see “[C.5.10 SMPP Server Options](#)” on page [980](#)

C.5.6

Configuring Mobile-to-Email Operation

To configure mobile-to-email functionality, two configuration steps must be performed:

- “[C.5.6.1 Configure a Mobile-to-Email Gateway Profile](#)” on page [972](#)
- “[C.5.6.2 Configure a Mobile-to-Email SMPP Server](#)” on page [972](#)

Note that multiple gateway profiles may use the same SMPP server instantiation. Indeed, the same SMPP server instantiation may be used for both email-to-mobile and mobile-to-email applications.

C.5.6.1

Configure a Mobile-to-Email Gateway Profile

For mobile origination, a gateway profile provides two key pieces of information: how to identify SMS messages intended for that profile and how to convert those messages to email messages. Note that this profile can be the same one used for email-to-mobile with the addition of the SELECT_RE option.

To configure the gateway profile, follow these steps:

▼ To Configure the Gateway Profile

1 Add a gateway profile (option group) to the SMS Gateway Server's configuration file.

To add an option group, use the following format:

```
[GATEWAY_PROFILE=profile_name]
option-name-1=option-value-1
option-name-2=option-value-2
...
option-name-n=option-value-n
```

Any name of 11 characters or less may be used for the profile's name. All that matters is that it is not already used for another gateway profile within the same configuration file.

2 Set the SELECT_RE option must be specified for each gateway profile.

The value of this option is an ASCII regular expression with which to compare SMS destination addresses. If an SMS destination address matches the regular expression, then the SMS message is sent through the gateway to email using the characteristics described by the matching profile.

It is important to note that it is possible to configure multiple gateway profiles which have overlapping sets of SMS addresses (for example, a profile which matches the address 000 and another which matches any other three-digit address). However, so doing should be avoided as an SMS message will be passed off to only one gateway profile: the first one which matches. And, moreover, the order in which they are compared is undefined.

3 Set the CHANNEL option.

Its value should be the name of the MTA's SMS channel.

For a complete description of all mobile origination options, see “[C.5.11 Gateway Profile Options](#)” on page [981](#)

C.5.6.2

Configure a Mobile-to-Email SMPP Server

Adding an SMPP server is the same as for the email-to-mobile SMPP server (see “[C.5.5.3 An SMPP Server](#)” on page [971](#)).

The remote SMSC needs to be configured to route SMS traffic to the gateway SMPP server. To do this, the SMS destination address used by the SMSC to route mobile-to-email traffic should be the value set for the gateway profile option SELECT_RE.

For example, if the SMS address 000 is to be used for mobile-to-email traffic, then the SMSC needs to be configured to route traffic for the SMS destination address 000 to the gateway SMPP server. The gateway profile should use the option setting SELECT_RE=000.

C.5.7 Configuration Options

The SMS Gateway Server configuration file options are detailed in this section. The tables that follow list all the available configuration options with a brief description of each. There is a table each for global options, SMPP relay options, SMPP server options, and SMS Gateway Server profile options.

In the subsections that follow, complete descriptions are given for all the available configuration options. The subsections are:

- “[C.5.8 Global Options](#)” on page 973

Global options must be placed at the top of the configuration file, before any option groups. The remaining options must appear within option groups.

- “[C.5.9 SMPP Relay Options](#)” on page 977
- “[C.5.10 SMPP Server Options](#)” on page 980
- “[C.5.11 Gateway Profile Options](#)” on page 981

C.5.8 Global Options

The SMS Gateway Server presently has three categories of global options:

- “[C.5.8.1 Thread Tuning Options](#)” on page 974
- “[C.5.8.2 Historical Data Tuning](#)” on page 975
- “[C.5.8.3 Miscellaneous](#)” on page 976

All global options must be specified at the top of the configuration file, before any option groups are specified. [Table C-20](#) lists all global configuration options.

TABLE C-20 Global Options

Options	Default	Description
“ DEBUG ” on page 976	6	Selects the types of diagnostic output generated
“ HISTORY_FILE_DIRECTORY ” on page 975		Absolute directory path for files of historical data
“ HISTORY_FILE_MODE ” on page 975	0770	Permissions for files of historical data

TABLE C-20 Global Options *(Continued)*

Options	Default	Description
“HISTORY_FILE_ROLLOVER_PERIOD” on page 975	30 mins	Maximum length of time to write to the same file of historical data
“LISTEN_CONNECTION_MAX” on page 977	10,000	Maximum number of concurrent inbound connections across all SMPP relay and server instantiations
“RECORD_LIFETIME” on page 975	3 days	Lifetime of a record in the historical data archive
“THREAD_COUNT_INITIAL” on page 974	10 threads	Initial number of worker threads
“THREAD_COUNT_MAXIMUM” on page 974	50 threads	Maximum number of worker threads
“THREAD_STACK_SIZE” on page 974	64 Kb	Stack size for each worker thread

C.5.8.1

Thread Tuning Options

Each inbound TCP connection represents an SMPP session. The processing for a session is handled by a worker thread from a pool of threads. When the session processing needs to wait for completion of an I/O request, the worker thread parks the session and is given other work to perform. When the I/O request completes, the session is resumed by an available worker thread from the pool.

The following options allow for tuning of this pool of worker thread processes:

[“THREAD_COUNT_INITIAL” on page 974](#), [“THREAD_COUNT_MAXIMUM” on page 974](#), [“THREAD_STACK_SIZE” on page 974](#).

THREAD_COUNT_INITIAL

(integer, > 0) Number of threads to initially create for the pool of worker threads. This count does not include the dedicated threads used to manage the in-memory historical data (2 threads) nor the dedicated threads used to listen for incoming TCP connections (one thread per TCP port/interface address pair the SMS Gateway Server listens on). The default value is for THREAD_COUNT_INITIAL is 10 threads.

THREAD_COUNT_MAXIMUM

(integer, >= THREAD_COUNT_INITIAL) Maximum number of threads to allow for the pool of worker threads. The default value is 50 threads.

THREAD_STACK_SIZE

(integer, > 0) Stack size in bytes for each worker thread in the pool of worker threads. The default value is 65,536 bytes (64 Kb).

C.5.8.2

Historical Data Tuning

When an SMS message is relayed, the message ID generated by the receiving, remote SMPP server is saved in an in-memory hash table. Along with this message ID, information about the original email message is also saved. Should that message ID subsequently be referenced by an SMS notification, this information may be retrieved. The retrieved information can then be used to send the SMS notification to the appropriate email recipient.

The in-memory hash table is backed to disk by a dedicated thread. The resulting disk files are referred to as “history files”. These history files serve two purposes: to save, in nonvolatile form, the data necessary to restore the in-memory hash table after a restart of the SMS Gateway Server, and to conserve virtual memory by storing potentially lengthy data on disk. Each history file is only written to for `HASH_FILE_ROLLOVER_PERIOD` seconds after which time it is closed and a new history file created. When a history file exceeds an age of `RECORD_LIFETIME` seconds, it is deleted from disk.

The following options allow for tuning historical files: “[HISTORY_FILE_DIRECTORY](#)” on page 975, “[HISTORY_FILE_MODE](#)” on page 975, “[HISTORY_FILE_ROLLOVER_PERIOD](#)” on page 975, “[RECORD_LIFETIME](#)” on page 975.

HISTORY_FILE_DIRECTORY

(*string, absolute directory path*) Absolute path to the directory to which to write the history files. The directory path will be created if it does not exist. The default value for this option is:

msg-svr-base/data/sms_gateway_cache/

The directory used should be on a reasonably fast disk system and have more than sufficient free space for the anticipated storage; see “[C.6 SMS Gateway Server Storage Requirements](#)” on page 989 to change this option to a more appropriate value.

HISTORY_FILE_MODE

(*integer, octal value*) File permissions to associate with the history files. By default, a value of 0770 (octal) is used.

HISTORY_FILE_ROLLOVER_PERIOD

(*integer, seconds*) The current history file is closed and a new one created every `HASH_FILE_ROLLOVER_PERIOD` seconds. By default, a value of 1800 seconds (30 minutes) is used.

RECORD_LIFETIME

(*integer, seconds > 0*) Lifetime in seconds of a historical record. Records older than this lifetime will be purged from memory; history files older than this lifetime will be deleted from disk. By default, a value of 259,200 seconds (3 days) is used. Records stored in memory are purged in

sweeps by a thread dedicated to managing the in-memory data. These sweeps occur every HASH_FILE_ROLLOVER_PERIOD seconds. Files on disk are purged when it becomes necessary to open a new history file.

C.5.8.3

Miscellaneous

These are miscellaneous options:

- “[DEBUG](#)” on page 976
- “[LISTEN_CONNECTION_MAX](#)” on page 977
- “[LOG_PAGE_COUNT](#)” on page 977

DEBUG

(*integer, bitmask*) Enable debug output. The default value is 6 which selects warning and error messages.

[Table C-21](#) defines the bit values of the DEBUG bitmask.

TABLE C-21 DEBUG Bitmask

Bit	Value	Description
0-31	-1	Extremely verbose output
0	1	Informational messages
1	2	Warning messages
3	4	Error messages
3	8	Subroutine call tracing
4	16	Hash table diagnostics
5	32	I/O diagnostics, receive
6	64	I/O diagnostics, transmit
7	128	SMS to email conversion diagnostics (mobile originate and SMS notification)
8	256	PDU diagnostics, header data
9	512	PDU diagnostics, body data
10	1024	PDU diagnostics, type-length-value data
11	2048	Option processing; sends all option settings to the log file.

LISTEN_CONNECTION_MAX

(*integer*, ≥ 0) The maximum number of concurrent, inbound TCP connections to allow across all SMPP relay and server instantiations. A value of 0 (zero) indicates that there is no global limit on the number of connections. There may, however, be per relay or server limits imposed by a given relay or server instantiation. Default: 10,000

LOG_PAGE_COUNT

(0, 1, 2) The LOG_PAGE_CONT SMS channel option only takes effect when logging is enabled for the channel with the `logging` channel keyword. When logging is enabled, this option controls the value recorded in the `mail.log` file's message size field. Normally that field gives the block size of the underlying message file. When LOG_PAGE_COUNT has a non-zero value, the number of transmitted pages will instead be recorded in that field of the log file.

0 - Log the block size of the underlying message file. This is the default behavior when LOG_PAGE_COUNT is not specified.

1 - Log the count of pages sent when the entire message is successfully transmitted to the recipient. Otherwise, log a page count of zero even if some pages were sent to the recipient.

2 - Log the count of pages sent to the recipient regardless of whether or not the entire message was sent.

The distinction between LOG_PAGE_COUNT=1 and LOG_PAGE_COUNT=2 is only relevant when a message is sufficiently large that it will be transmitted as several pages. In that case, it is possible that before transmitting all the pages an error might occur. For example, the network between the MTA and the remote SMPP server goes down. In that case, a later retransmission attempt will be made for the message. For each attempt, the previously sent pages are sent again along with the pages which were not sent. Sites may choose whether or not they want to record the count of pages successfully sent during these failed delivery attempts.

C.5.9

SMPP Relay Options

The SMS Gateway Server can have multiple instantiations of its SMPP relay, each with different characteristics chief of which will be the TCP port and interface listened on. Put differently, for each network interface and TCP port pair the SMPP relay listens on, distinct characteristics may be ascribed. These characteristics are specified using the options described in this section.

Each instantiation should be placed within an option group of the form:

```
[SMPP_RELAY=relay-name]
option-name-1=option-value-1
option-name-2=option-value-2
...
option-name-n=option-value-n
```

The string `relay-name` merely serves to differentiate this instantiation from other instantiations.

Table C–22 lists the SMPP relay configuration options.

TABLE C–22 SMPP Relay Options

Options	Default	Description
“C.5.9.1 LISTEN_BACKLOG” on page 978	255	Connection backlog for inbound SMPP client connections
“LISTEN_CONNECTION_MAX” on page 978		Maximum number of concurrent inbound connections
“LISTEN_INTERFACE_ADDRESS” on page 979		Network interface for inbound SMPP client connections
“LISTEN_PORT” on page 979		TCP port for inbound SMPP client connections
“LISTEN_RECEIVE_TIMEOUT” on page 979	600 s	Read timeout for inbound connections from SMPP clients
“LISTEN_TRANSMIT_TIMEOUT” on page 979	120 s	Write timeout for inbound connections from SMPP clients
“MAKE_SOURCE_ADDRESSES_UNIQUE” on page 979	1	Make relayed SMS source addresses unique and able to be replied to
“SERVER_HOST” on page 979		Host name or IP address of the SMPP server to relay to
“SERVER_PORT” on page 979		TCP port of the SMPP server to relay to
“SERVER_RECEIVE_TIMEOUT” on page 980	600 s	Read timeout for outbound SMPP server connections
“SERVER_TRANSMIT_TIMEOUT” on page 980	120 s	Write timeout for outbound SMPP server connections

C.5.9.1

LISTEN_BACKLOG

(integer, in [0, 255]) Connection backlog allowed by the TCP stack for inbound SMPP client connections. The default value is 255.

LISTEN_CONNECTION_MAX

(integer, >= 0) The maximum number of concurrent, inbound TCP connections to allow for this SMPP relay instantiation. Note that this value will be ignored if it exceeds the global LISTEN_CONNECTION_MAX setting.

LISTEN_INTERFACE_ADDRESS

(string, "INADDR_ANY" or dotted decimal IP address) The IP address of the network interface to listen for to inbound SMPP client connections. May be either the string "INADDR_ANY" (all available interfaces) or an IP address in dotted decimal form. (For example, 193.168.100.1) The default value is "INADDR_ANY". Clustered HA configurations will need to set this value to correspond to the HA logical IP address.

LISTEN_PORT

(integer, TCP port number) TCP port to bind to for accepting inbound SMPP client connections. Specification of this option is mandatory; there is no default value for this option. Note also that there is no Internet Assigned Numbers Authority (IANA) assignment for this service.

LISTEN_RECEIVE_TIMEOUT

(integer, seconds > 0) Timeout to allow when waiting to read data from an SMPP client. The default value is 600 seconds (10 minutes).

LISTEN_TRANSMIT_TIMEOUT

(integer, seconds > 0) Timeout to allow when sending data to an SMPP client. The default value is 120 seconds (2 minutes).

MAKE_SOURCE_ADDRESSES_UNIQUE

(0 or 1) By default, the SMPP relay will append to each SMS source address a unique, ten digit string. The resulting SMS source address is then saved along with the other historical data. The result is a unique SMS address which may then be replied to by SMS users. The SMPP server will detect this address when used as an SMS destination address and will then send the SMS message to the correct email originator.

To disable this generating of unique SMS source addresses (for one-way SMS), specify a value of 0 (zero) for this option.

SERVER_HOST

(string, TCP hostname or dotted decimal IP address) SMPP server to relay SMPP client traffic to. Either a hostname or IP address may be specified. Specification of this option is mandatory; there is no default value for this option.

SERVER_PORT

(integer, TCP port number) TCP port for the remote SMPP server to which to relay. Specification of this option is mandatory; there is no default value for this option. There is no IANA assignment for this service; do not confuse with the IANA assignment for SNPP.

SERVER_RECEIVE_TIMEOUT

(*integer, seconds > 0*) Timeout to allow when waiting to read data from the SMPP server. The default value is 600 seconds (10 minutes).

SERVER_TRANSMIT_TIMEOUT

(*integer, seconds > 0*) Timeout to allow when sending data to the SMPP server. The default value is 120 seconds (2 minutes).

C.5.10 SMPP Server Options

The SMS Gateway Server can have multiple instantiations of its SMPP server, each with different characteristics chief of which will be the TCP port and interface listened on. Put differently, for each network interface and TCP port pair the SMPP server listens on, distinct characteristics may be ascribed. These characteristics are specified using the options described in this section.

Each instantiation should be placed within an option group of the form:

```
[SMPP_SERVER=server-name]
option-value-1=option-value-1
option-value-2=option-value-2
...
option-name-n=option-value-n
```

The string `server-name` merely serves to differentiate the instantiation from other instantiations.

[Table C-23](#) lists the SMPP server configuration options.

TABLE C-23 SMPP Server Options

Options	Default	Description
“C.5.10.1 LISTEN_BACKLOG” on page 981	255	Connection backlog for inbound SMPP server connections
“LISTEN_CONNECTION_MAX” on page 981		Maximum number of concurrent inbound connections
“LISTEN_INTERFACE_ADDRESS” on page 981		Network interface for inbound SMPP server connections
“LISTEN_PORT” on page 981		TCP port for inbound SMPP server connections

TABLE C-23 SMPP Server Options *(Continued)*

Options	Default	Description
“LISTEN_RECEIVE_TIMEOUT” on page 981	600 s	Read timeout for inbound SMPP server connections
“LISTEN_TRANSMIT_TIMEOUT” on page 981	120 s	Write timeout for inbound SMPP server connections

C.5.10.1

LISTEN_BACKLOG

(integer in [0,255]) Connection backlog allowed by the TCP stack for inbound SMPP client connections. The default value is 255.

LISTEN_CONNECTION_MAX

(integer >= 0) The maximum number of concurrent, inbound TCP connections to allow for this SMPP server instantiation. Note that this value will be ignored if it exceeds the global LISTEN_CONNECTION_MAX setting.

LISTEN_INTERFACE_ADDRESS

(string, “INADDR_ANY” or dotted decimal IP address) The IP address of the network interface to listen to for inbound SMPP client connections on. May be either the string “INADDR_ANY” (all available interfaces) or an IP address in dotted decimal form. (For example, 193.168.100.1.) The default value is “INADDR_ANY”.

LISTEN_PORT

(integer, TCP port number) TCP port to bind to for accepting inbound SMPP client connections. Specification of this option is mandatory; there is no default value for this option. Note that there is no IANA assignment for this service.

LISTEN_RECEIVE_TIMEOUT

(integer, seconds > 0) Timeout to allow when waiting to read data from an SMPP client. The default value is 600 seconds (10 minutes).

LISTEN_TRANSMIT_TIMEOUT

(integer, seconds > 0) Timeout to allow when sending data to an SMPP client. The default value is 120 seconds (2 minutes).

C.5.11

Gateway Profile Options

There may be zero or more gateway profiles. In the SMS Gateway Server’s configuration file, each gateway profile is declared within an option group in the following format:

```
[GATEWAY_PROFILE=profile-name]
option-name-1=option-value-1
option-name-2=option-value-2
...
option-name-n=option-value-n
```

The string `profile-name` merely serves to differentiate the profile from other origination profiles.

[Table C-24](#) lists the SMS Gateway Server profile options.

TABLE C-24 SMS Gateway Server Profile Options

Options	Default	Description
“C.5.11.1 CHANNEL” on page 982	sms	Channel to enqueue message as
“EMAIL_BODY_CHARSET” on page 982	US-ASCII	Character set for email message bodies
“EMAIL_HEADER_CHARSET” on page 983	US-ASCII	Character set for email message headers
“FROM_DOMAIN” on page 983		Domain name for routing email back to SMS
“PARSE_RE_0, PARSE_RE_1, ..., PARSE_RE_9” on page 983		Regular expressions for parsing SMS message text
“PROFILE” on page 985	GSM	SMS profile to operate under: GSM, TDMA, or CDMA
“SELECT_RE” on page 985		Regular expression for selecting the plugin
“SMSC_DEFAULT_CHARSET” on page 985	US-ASCII	SMSC's default character set
“USE_SMS_PRIORITY” on page 985	0	Gateway SMS priority flags to email
“USE_SMS_PRIVACY” on page 986	0	Gateway SMS privacy indicators to email

C.5.11.1 CHANNEL

(*string, 1-40 characters*) Name of the MTA channel used to enqueue email messages. If not specified, then “sms” is assumed. The specified channel must be defined in the MTA’s configuration.

EMAIL_BODY_CHARSET

(*string, character set name*) The character set to translate SMS text to prior to insertion into an email message’s body. If necessary, the translated text will be MIME encoded. The default value is US-ASCII. If the SMS message contains glyphs not available in the charset, they will be converted to mnemonic characters, which may or may not be meaningful to the recipient.

A list of the character sets known to the MTA may be found in the following file:

installation-directory/config/charsets.txt

EMAIL_HEADER_CHARSET

(*string, character set name*) The character set to translate SMS text to prior to insertion into an RFC 822 Subject: header line. If necessary, the translated string will be MIME encoded. The default value is US-ASCII. If the SMS message contains glyphs not available in the charset, they will be converted to mnemonic characters, which may or may not be meaningful to the recipient.

FROM_DOMAIN

(*string, IP host name, 1-64 characters*) Domain name to append to SMS source addresses when constructing envelope From: addresses for email messages. The name specified should be the correct name for routing email back to SMS. (For example, the host name associated with the MTA SMS channel.) If not specified, then the official host name of the channel specified with the CHANNEL option will be used.

PARSE_RE_0, PARSE_RE_1, ..., PARSE_RE_9

(*string, UTF-8 regular expression*) For mobile origination of email, the gateway profile needs to extract a destination email address from the text of the SMS message. This is done by means of one or more POSIX-compliant regular expressions (REs). The text of the SMS message will be evaluated by each regular expression until either a match producing a destination email address is found or the list of regular expressions exhausted.

Note – Use of PARSE_RE_* and ROUTE_TO options are mutually exclusive. Use of both in the same gateway profile is a configuration error.

Each regular expression must be POSIX compliant and encoded in the UTF-8 character set. The regular expressions must output as string 0 the destination address. They may optionally output text to use in a Subject: header line as string 1, and text to use in the message body as string 2. Any text not “consumed” by the regular expression will also be used in the message body, following any text output as string 2.

The regular expressions will be tried in the order PARSE_RE_0, PARSE_RE_1, ..., up to PARSE_RE_9. If no regular expressions are specified, then the following default regular expression is used:

```
[ \t]*([^\(\ )]*[ \t]*(?:\(((^\ )*)\))?[ \t]*(.*))
```

This default regular expression breaks into the following components:

```
[ \t]*
```

Ignore leading white space characters (SPACE and TAB).

([^\\(]*)

Destination email address. This is the first reported string.

[\\t]*

Ignore white space characters.

(?:\\((([^\\])*)\\))?

Optional subject text enclosed in parentheses. This is the second reported string. The leading ?: causes the outer parentheses to not report a string. They are being used merely for grouping their contents together into a single RE for the trailing ?. The trailing ? causes this RE component to match only zero or one time and is equivalent to the expression {0,1}.

[\\t]*

Ignore white space characters.

(.*)

Remaining text to message body. This is the third reported string.

For example, with the above regular expression, the sample SMS message:

dan@sesta.com(Testing)This is a test

yields the email message:

To: dan@sesta.com

Subject: Testing

This is a test

As a second example, the SMS message:

sue@sesta.com This is another test

would yield:

To: sue@sesta.com

This is another test

Note that the SMS message, prior to evaluation with these regular expressions, will be translated to the UTF-16 encoding of Unicode. The translated text is then evaluated with the regular expressions which were previously converted from UTF-8 to UTF-16. The results of the evaluation are then translated to US-ASCII for the destination email address, EMAIL_HEADER_CHARSET for the Subject: text, if any, and EMAIL_BODY_CHARSET for the message body, if any.

PROFILE

(string, “GSM”, “TDMA”, or “CDMA”) SMS profile to assume. Presently this information is only used to map SMS priority flags to RFC 822 Priority: header lines. Consequently, this option has no effect when `USE_SMS_PRIORITY=0` which is the default setting for that option.

SELECT_RE

(string, US-ASCII regular expression) A US-ASCII POSIX-compliant regular expression to compare against each SMS message’s SMS destination address. If an SMS message’s destination address matches this RE, then the SMS message will be sent through the gateway to email in accord with this gateway profile.

Note that since an SMS message’s destination address is specified in the US-ASCII character set, this regular expression must also be expressed in US-ASCII.

SMSC_DEFAULT_CHARSET

(string, character set name) The name of the default character set used by the remote SMSC. The two common choices for this option are US-ASCII and UTF-16-BE (USC2). If not specified, US-ASCII is assumed.

USE_SMS_PRIORITY

(integer, 0 or 1) By default (with `USE_SMS_PRIORITY=0`), priority flags in SMS messages are ignored and not sent with the email messages. To have the priority flags passed with the email, specify `USE_SMS_PRIORITY=1`. When passed with the email, the mapping from SMS to email is as shown in [Table C-25](#):

TABLE C-25 Priority Flag Mapping from SMS to Email

SMS Profile	SMS Priority Flag	Email Priority: Header Line
GSM	0 (Non-priority)	No header line (implies Normal)
	1, 2, 3 (Priority)	Urgent
TDMA	0 (Bulk)	Nonurgent
	1 (Normal)	No header line (implies Normal)
	2 (Urgent)	Urgent
	3 (Very Urgent)	Urgent

TABLE C-25 Priority Flag Mapping from SMS to Email *(Continued)*

SMS Profile	SMS Priority Flag	Email Priority: Header Line
CDMA	0 (Normal)	No header line (implies Normal)
	1 (Interactive)	Urgent
	2 (Urgent)	Urgent
	3 (Emergency)	Urgent

Note that the email Priority: header line values are Nonurgent, Normal, and Urgent.

USE_SMS_PRIVACY

(integer, 0 or 1) By default (with USE_SMS_PRIVACY=0), SMS privacy indications are ignored and not sent with the email messages. To have this information passed with the email, specify USE_SMS_PRIVACY=1. When passed along with email, the mapping from SMS to email is as shown in [Table C-26](#):

TABLE C-26 Privacy Flags Mapping from SMS to Email

SMS Privacy Flag	Email Sensitivity: Header Line
0 (Not restricted)	No header line
1 (Restricted)	Personal
2 (Confidential)	Private
3 (Secret)	Company-confidential

Note that the values of the email Sensitivity: header line are Personal, Private, and Company-confidential.

C.5.12 Configuration Example for Two-Way SMS

Assumptions on Behavior

For the sake of this example, let us assume that the following behavior is desired:

- Email messages addressed to
`sms-id@sms.domain.com`
 are to be sent to the SMS address
`sms-id`
 and given a unique SMS source address in the range `000nnnnnnnnn`.

- Mobile SMS messages addressed to the SMS address **000** are to be sent through the gateway to email with the email address extracted from the start of the SMS message text.

For example, if the SMS message text is:

```
jdoe@domain.com Interested in a movie?
```

then the message “Interested in a movie?” is to be sent to jdoe@domain.com.

- SMS notifications sent to **000nnnnnnnnn** are to be sent through the gateway to email and directed to the originator of the message being receipted.

In order to bring about this behavior, the following assumptions and assignments are made

Further Assumptions and Assignments

- The MTA’s SMS channel uses the domain name **sms.domain.com**.
- The SMS Gateway Server runs on the host **gateway.domain.com** and uses:
 - TCP port 503 for its SMPP relay
 - TCP port 504 for its SMPP server
- The remote SMSC’s SMPP server runs on the host **smpp.domain.com** and listens on TCP port 377.
- The remote SMSC’s default character set is UCS2 (aka, UTF-16).

SMS Channel Configuration

To effect the above behavior, the following SMS channel configuration may be used in the **imta.cnf** file (add these lines to the bottom of the file):

```
(blank line)
sms
sms.domain.com
```

SMS Channel Option File

The channel’s option file, **sms_option**, would then contain the following settings:

```
SMPP_SERVER=gateway.domain.com
SMPP_PORT=503
USE_HEADER_FROM=0
DEFAULT_SOURCE_ADDRESS=000
GATEWAY_PROFILE=sms1
SMSC_DEFAULT_CHARSET=UCS2
```

SMS Gateway Server Configuration

Finally, the Gateway Server configuration file, **sms_gateway.cnf**, should look like the following:

```
HISTORY_FILE_DIRECTORY=/sms_gateway_cache/
[SMPP_RELAY=relay1]
LISTEN_PORT=503SERVER_HOST=smpp.domain.com
SERVER_PORT=377

[SMPP_SERVER=server1]
LISTEN_PORT=504

[GATEWAY_PROFILE=sms1]
SELECT_RE=000([0-9]{10,10}){0,1}
SMSC_DEFAULT_CHARSET=UCS2
```

Testing This Configuration

If you do not have an SMSC to test on, you may want to perform some loopback tests. With some additional settings in the `sms_option` file, some simple loop back tests may be performed for the above configuration.

C.5.12.1

Additional `sms_option` File Settings

The additional settings for the `sms_option` file are:

```
! So that we don't add text to the body of the SMS message
FROM_FORMAT=
SUBJECT_FORMAT=
CONTENT_PREFIX=
```

Without these settings, an email containing:

```
user@domain.com (Sample subject) Sample text
```

would get converted into the SMS message:

```
From:user@domain.com Subject:Sample Subject Msg:Sample text
```

That, in turn, would not be in the format expected by the mobile-to-email code, which wants to see:

```
user@domain.com (Sample subject) Sample text
```

Hence the need (for loopback testing) to specify empty strings for the `FROM_FORMAT`, `SUBJECT_FORMAT`, and `CONTENT_PREFIX` options.

Performing the Loopback Test

Send test email messages addressed to `000@sms.domain.com`, such as:

```
user@domain.com (Test message) This is a test message which should loop back
```

The result is that this email message should be routed back to the email recipient user@domain.com. Be sure to have added sms.domain.com to your DNS or host tables for the test.

C.6 SMS Gateway Server Storage Requirements

To determine the amount of resources you will need for the SMS Gateway Server, use the numbers you generate from the requirements in [Table C-27](#) along with your expected number of relayed messages per second and the RECORD_LIFETIME setting.

[Table C-27](#) covers the requirements for the historical records, the SMPP relay, and SMPP server.

TABLE C-27 SMS Gateway Server Storage Requirements

Component	Requirements
In-memory historical record	<p>Each relayed message requires $33+m+s$ bytes of virtual memory, where m is the length of the message's SMS message ID ($1 \leq m \leq 64$) and s is the length of the message's SMS source address ($1 \leq s \leq 20$).</p> <p>When <code>MAKE_SOURCE_ADDRESS_UNIQUE=0</code>, then only $16+m$ bytes are used. For 64 bit operating systems, $49+m+s$ bytes of virtual memory are consumed per record [$24+m$ when <code>MAKE_SOURCE_ADDRESS_UNIQUE=0</code>].</p> <p>Note also, that the heap allocator may actually allocate larger size pieces of virtual memory for each record.</p> <p>The maximum number of records is 43 billion ($2^{32}-1$). For less than 16.8 million records (2^{24}), the hash table consumes approximately 16 Mb; for less than 67.1 million records (2^{26}), the hash table consumes approximately 64 Mb; for more than 67.1 million records, the hash table consumes approximately 256 Mb.</p> <p>Double the memory consumptions for 64 bit operating systems.</p> <p>These consumptions are in addition to the memory consumption required for each record itself.</p>

TABLE C-27 SMS Gateway Server Storage Requirements *(Continued)*

Component	Requirements
On-disk historical record	<p>Each relayed message requires on average the following number of bytes:</p> $81 + m + 2s + 3a + S + 2i$ <p>where:</p> <ul style="list-style-type: none"> ■ m is the average length of SMS message IDs, and $1 \leq m \leq 64$ ■ s is the average length of SMS source addresses, and $1 \leq s \leq 20$ ■ a is the average length of email addresses, and $3 \leq a \leq 129$ ■ S is the average length of <code>Subject</code>: header lines, and $0 \leq S \leq 80$ ■ i is the average length of email message envelope IDs, and $0 \leq i \leq 129$ <p>The size for any specific record is influenced by the length of the message's envelope <code>From</code>: and <code>To</code>: addresses, envelope and message IDs, and the length of the <code>Subject</code>: header line.</p> <p>The maximum record length is 910 bytes.</p> <p>When <code>MAKE_SOURCE_ADDRESS_UNIQUE=0</code> is used, the size of each record in bytes is: $78 + m + 3a + S + 2i$.</p>
SMPP relay	Each relayed SMPP session consumes two TCP sockets: one with the local SMPP client and another with the remote SMPP server. Approximately 1 Kb of virtual memory is consumed per connection on 32 bit operating systems; 2 Kb on 64 bit operating systems.
SMPP server	Each incoming connection consumes a TCP socket. Approximately 1 Kb of virtual memory is consumed per connection on 32 bit operating systems; 2 Kb on 64 bit operating systems.

For instance, if on average 50 messages per second are expected to be relayed, SMS source addresses are 13 bytes long, SMS message IDs have a typical length of 12 bytes, email addresses 24 bytes, `Subject`: lines 40 bytes, email message and envelope IDs 40 bytes each, and historical data is retained for 7 days, then:

- There will be 30.24 million historical records to store, each on average 58 bytes in memory and 311 bytes long on disk;
- The in-memory consumption of the historical records will be about 1.70 Gb (1.63 Gb + 64 Mb); and
- The on-disk storage will be approximately 8.76 Gb.

While a sufficiency of disk may be supplied to handle any on disk requirements, the virtual memory requirement on a 32-bit machine will be a hard limit of approximately 2 Gb. To reduce the amount of virtual memory or disk storage required, use the `RECORD_LIFETIME` option to reduce the length of time records are retained.

Installation Worksheets

This appendix provides worksheets by which you can plan your installation. The following worksheets are included:

- “[D.1 Directory Server Installation](#)” on page 991
- “[D.2 Directory Server Setup Script \(comm_dssetup.pl\)](#)” on page 993
- “[D.3 Messaging Server Initial Runtime Configuration](#)” on page 994

D.1 Directory Server Installation

You installed Directory Server through the Java Enterprise System installer or through a previous installation. Record your Directory Server installation and configuration parameters in [Table D-1](#) (this is a replica of the worksheet shown in the Communications Suite Deployment Planning Guide). You will need these parameters when you install and configure the Administration and Messaging servers.

TABLE D-1 Directory Server Installation Parameters

Parameter:	Description:	Example:	Used in:	Your Answers:
Directory Installation Root	A directory on the directory server machine dedicated to holding the server program, configuration, maintenance, and information files.	/var/mps/serverroot/	comm_dssetup.pl Perl script See “ 1.2 To Prepare Directory Server for Messaging Server Configuration ” on page 64	

TABLE D-1 Directory Server Installation Parameters *(Continued)*

Parameter:	Description:	Example:	Used in:	Your Answers:
Host	The host name is the IP host name, which might be either a “short-form” host name (for example, <code>fiddle</code>) or a fully qualified host name. The fully qualified host name consists of two parts: the host name and the domain name.	<code>fiddle.west.sesta.com</code>	Administration Server Configuration See “1.2 To Prepare Directory Server for Messaging Server Configuration” on page 64	
LDAP Directory Port Number	The default for and LDAP directory server is 389.	389	Administration Server Configuration and Messaging Server Configuration See “1.2 To Prepare Directory Server for Messaging Server Configuration” on page 64 and “1.3 Creating the Initial Messaging Server Runtime Configuration” on page 65.	
Administrator ID and Password	Administrator in charge or responsible for configuration information. Password for the Administrator	Admin PaSsWoRd	Administration Server Configuration See “1.2 To Prepare Directory Server for Messaging Server Configuration” on page 64	
User and Group Tree Suffix	The distinguished name of the LDAP entry at the top of the directory tree, below which user and group data is stored.	<code>o=usergroup</code>	<code>comm_dssetup.pl</code> Perl script See “1.2 To Prepare Directory Server for Messaging Server Configuration” on page 64	

TABLE D-1 Directory Server Installation Parameters *(Continued)*

Parameter:	Description:	Example:	Used in:	Your Answers:
Directory Manager DN and Password	The privileged directory administrator, comparable to the superuser user in UNIX. Typically, this administrator is responsible for user and group data. Password for the Directory Manager.	cn=Directory Manager pAsSwOrD	comm_dssetup.pl Perl script and Messaging Server Configuration	See “1.2 To Prepare Directory Server for Messaging Server Configuration” on page 64 and “1.3 Creating the Initial Messaging Server Runtime Configuration” on page 65.
Administration Domain	A region of administrative control.	System Lab	Administration Server Configuration	See “1.2 To Prepare Directory Server for Messaging Server Configuration” on page 64

D.2 Directory Server Setup Script (comm_dssetup.pl)

When you run the Directory Server Setup script (`comm_dssetup.pl`) to prepare Directory Server for Messaging Server configuration, record your installation parameters in **Table D-2**. You will need some of these parameters for the Messaging Server initial runtime configuration.

TABLE D-2 `comm_dssetup.pl` Script Parameters

Parameter	Description	Example	Your Answers:
Server Root	Installation Root of the Directory Server dedicated to holding the server program, configuration, maintenance, and information files.	/var/mps/serverroot/	

TABLE D-2 comm_dsetup.pl Script Parameters (*Continued*)

Parameter	Description	Example	Your Answers:
Server Instance	LDAP Directory Server daemon or service that is responsible for most functions. In certain deployments, you might dedicate an instance for maintaining users and groups and maintain a separate instance for configuration.		
DC Root	If you want to have a two-tree DIT provisioning model (Sun LDAP Schema 1 or Sun ONE LDAP Schema.2 compatibility mode), the DC Tree mirrors the local DNS structure and is used by the system as an index to the Organization Tree that contain the user and group data entries.	o=internet	
User and Group Base Suffix	Top entry in the Organization Tree which holds the namespace for user and group entries.	o=usergroup	
Directory Manager DN and Password	Administrator who is responsible for the user and group data in the Organization Tree. Should be the same as what was specified in the Sun Java Enterprise System Installer. Password of Directory Manager DN	cn=Directory Manager pAsSwOrD	

D.3 Messaging Server Initial Runtime Configuration

When you run the Messaging Server initial runtime configuration program, record your installation parameters in [Table D-3](#). You might also refer to your “[D.1 Directory Server Installation](#)” on page [991](#) checklist to answer certain questions.

TABLE D–3 Initial Runtime Configuration Parameters

Parameter	Description	Example	Your Answers:
Configuration and Data Directory	Contains all of the Messaging Server configuration files. The <i>msg-svr-base/data</i> directory is symbolically linked to this directory.	/var/mps/SUNmsgsr/ 	
UNIX System User	Certain privileges designated to system users to ensure they have appropriate permissions for the processes they are running. This system user should not be the same user that you specified in the Administration Server Initial Runtime Configuration.	mailsrv	
UNIX System Group	The group to which certain UNIX System users belong. This system group should not be the same group that you specified in the Administration Server Initial Runtime Configuration.	mail	
Configuration Directory LDAP URL, Directory Manager, and Password	Configuration Directory Server, LDAP URL, Bind DN, and Password	ldap://fiddle.west.sesta.com:389 cn=Directory Manager PaSsWoRd	
User and Group Directory LDAP URL, Directory Manager, and Password	User and Group Directory Server, LDAP URL, Bind DN, and Password. It is recommended that you have a separate User and Group directory from your Configuration directory.	ldap://fiddle.west.sesta.com:389 cn=Directory Manager PaSsWoRd	
Postmaster Email Address	Email address of the administrator who will monitor postmaster mail. Address must be a fully qualified address, and must be valid, in that there is a mailbox associated with the address.	pma@siroe.com	

TABLE D-3 Initial Runtime Configuration Parameters *(Continued)*

Parameter	Description	Example	Your Answers:
Password for Administrator Accounts	Password that will be used for service administrator, user/group administrator, end user administrator privileges as well as PAB administrator and SSL passwords.	paSSwoRD	
Default Email Domain	The email default that is used if no domain is specified.	siroe.com	
Organization Name for Default Email Domain	An organization name under which your organization will reside and is used to construct the organization tree.	For example if your organization name is Engineering, all the users for siroe.com (your default email domain) will be placed under the LDAP DN o=Engineering, o=usergroup Your user and group directory suffix was specified in comm_dsetup.pl.	

Glossary

Glossary

Refer to the [*Sun Java Enterprise System Glossary*](#) for a complete list of terms that are used in this documentation set.

Index

Numbers and Symbols

*[,], 672
+, 140
\$?[,], 313
\! (exclamation point), in addresses, 299
\| vertical bar, 294
@ (at sign), 313
! (exclamation point), as a comment indicator, 236
< (less than sign), including files with, 236
% (percent sign), 310
\$A, 312
(A\\!B)%C, 391
\$B, 311
\$C, 310-311, 313
\$E, 311
\$F, 311
\$M, 310, 313
/ matching, 243
\$N, 310, 313
\$P, 312
\$Q, 310-311, 313
\$R, 212-213, 311
\$S, 312
\$T, 313
\$U substitution sequence, 302
\$V, 206
\$V Metacharacter, 210-211
\$X, 312
\$Z, 206
120230-08, 671-672
220 banner, 848
733, 390

8-bit characters, 859

822, 390

A

A\\!(B%C), 391
A!B%C, 391
A!B@C, 391
A@B@C, 392
acceptalladdresses, 400
acceptvalidaddresses, 400
access control
See also mapping tables
access to TCP services, overview, 737-746
client access, 145
creating access filters, 745
filter syntax, 739-743
HTTP service, 145, 737-746
IMAP service, 145, 737-746
mapping tables, 542
message store, 594-596
monitoring users, 659-660
POP service, 145, 737-746
SMTP service, 542
testing mappings, 556-557
when applied, 556
Access Manager, 157
ACCESS_ORCPT, 547, 548
action, 629
address
! and % usage, 391

address (*Continued*)

- blank envelope return, 280-281
- destination, 416
- handling, 389-400
- incomplete, 393-394
- interpretation, 391
- multiple destination, 415
- rewriting, 393
- routing information, 391-392
- address changing, 264
- Address in Received\\, header, 396
- address mapping, FORWARD, 268-271
- address message headers
 - comments in, 396-397
 - personal names, 397-398
- address reversal, 228-230
- address-reversal database, 264
- Address-Reversal database, 264
- address reverse, channel-specific, 267
- address reverse controls, 266
- address rewriting, 393
- addresses
 - backward-pointing, 392
 - envelope To\\, 311
 - From\\, 392
 - interpreting, 391
 - invalid, 280
- addressssrs, 519
- addreturnpath, 396
- addrspersfile, 415
- Admin Console, 124
- administrator access control
 - configuring, 735-737
 - to message store, 594
 - to server as a whole, 736
 - to server tasks, 736-737
- after channel keyword, 381
- AgentX protocol, 895-896
- AGIC, 209
- aging policies
 - See* automatic message removal
 - message store, 625-635
 - number of messages, 625
 - size of mailbox, 625

aging policies (*Continued*)

- specifying, 625-635
- alarm attributes, disk space, 642
- alarm.diskavail, 889
- alarm.diskavail.msgalarmdescription, 870
- alarm.diskavail.msgalarmstatinterval, 870, 889
- alarm.diskavail.msgalarmthreshold, 870, 889
- alarm.diskavail.msgalarmthresholddirection, 889
- alarm.diskavail.msgalarmwarninginterval, 870, 889
- alarm.msgalarmnoticehost, 889
- alarm.msgalarmnoticeport, 889
- alarm.msgalarmnoticercpt, 871, 889
- alarm.msgalarmnoticesender, 889
- alarm.serverresponse, 889
- alarm.serverresponse.msgalarmstatinterval, 889
- alarm.serverresponse.msgalarmthreshold, 889
- alarm.serverresponse.msgalarmthresholddirection, 889
- alarm.serverresponse.msgalarmwarninginterval, 889
- alias database, 398
- ALIAS_DOMAINS, 398
- ALIAS_ENTRY_CACHE_SIZE, 227
- ALIAS_ENTRY_CACHE_TIMEOUT, 227
- alias expansion, 209
- alias file, 398
- ALIAS_MAGIC, enabling direct LDAP, 231-232
- ALIAS_URL0, 209
 - enabling direct LDAP, 231-232
- ALIAS_URL1, 209
 - enabling direct LDAP, 231-232
- ALIAS_URL2, 209
 - enabling direct LDAP, 231-232
- aliasdetourhost, 421
- aliasedObjectName, 207
- Aliases, 261
- aliases
 - alias database, 262
 - alias file, 252, 262
 - including other files in aliases file, 263
- aliases file, 271
- aliaslocal, 398
- aliasoptindetourhost, 421
- aliaspostmaster, 281
- ALLOW_RECIPIENTS_PER_TRANSACTION, 360
- ALLOW_REJECTIONS_BEFORE_DEFERRAL, 425

-
- ALLOW_TRANSACTIONS_PER_SESSION, 360
 allowetrn, 364
 allowetrn channel keyword, 364
 allowswitchchannel channel keyword, 374
 altered addresses in notification messages, 279
 alternate channel for incoming mail, 374-375
 alternate conversion channel, 421
 alternateblocklimit, 412-413
 alternatetechannel, 412-413
 alternatelinelimit, 412-413
 alternaterecipientlimit, 412-413
 alwaysencrypt, 764
 alwayssign, 764
 AMSDK, 159
 anti-spam, 463, 493, 541, 625-635
Anti-Spam, 509-510
 anti-spam
 actions, 471
 Brightmail
 See Brightmail
 Channel-level Filtering, 469-470, 470
 Client Library, 465-466
 Anti-spam, Cloudmark, 509
 anti-spam
 deploying 3rd-party software, 464
 Domain-level Filtering, 468
 Library Paths, 465-466
 limiting recipients, 414-415
 messages to filter, 466
 multiple programs, 465-466
 Sieve, 471
 spam score, 463, 493
 SpamAssassin
 See SpamAssassin
 Theory of Operations, 464
 User-level Filtering, 466-467
 anti-virus, 463, 476, 493
 scanner, 421
 APOP, 717
 appid, 168
 Application ID, 160
 Arabic Character Detection, 450-451
 archiving, 687
 associatedDomain, 207
 asterisks, 859
 asterisks, in addresses, 194
 at sign, 299, 310, 313
 attachments, 405-410
 opening, 443
 authenticated addresses, 377-378
 authentication
 certificate-based, 715, 720
 HTTP, 140-142
 IMAP, 140-142
 mechanisms, 715
 Messaging Multiplexor, 175
 password, 719
 POP, 140-142
 SASL, 715
 SMTP, 720
 authrewrite, 377
 auto_ef, 450
 automatic fragmentation of large messages, 408-409
 automatic message removal, 625-635
 by message type, 614-616
 exclude users, 626, 635
 localized filepatterns, 630-631
 policy definition, 626-627, 632-633
 rule setting, 627-633
 scheduling, 633
 automatic restart, 127
 automatic restart, high availability, 129
 automatic task scheduling, 129-131
 autoreply, 533
 autoreply, forwarded messages, 539-540
 autoreply caching, 408

B

- backoff, 383
 backoff channel keyword, 381
 backup groups, 648
 backup procedure for message store
 backup utilities, 649
 creating a policy, 647
 creating backup groups, 648
 description, 646
 full backup, 648

backup procedure for message store (*Continued*)

- incremental backup, 648
- parallel backups, 648
- peak business loads, 648
- serial backups, 648
- single copy procedure, 647
- using Legato Networker, 653
- using third party software, 656

backward-pointing addresses, 392

bad equivalence for alias, MTA error messages, 860

bang-style (UUCP) addresses, 293

bang-style address conventions, 299

bangoverpercent, 391

bangoverpercent keyword, 298

bangstyle, 390

banners

- IMAP, 139

- POP, 139

base63, 410

bidirectional, 382

bit flags, 281, 283

blank envelope addresses, 281, 283

blank envelope return addresses, 280-281

blank lines, in a configuration file, 236

BLOCK_SIZE, 408, 411

blocketrn, 364

blocketrn channel keyword, 364

blocklimit, 411

blSWClientDesintationForeign, 480

blSWClientDestinationDefault, 480

blSWClientDestinationLocal, 480

blswcServerAddress, 480

blSWLocalDomain, 480

blSWPrecedence, 479

blSWUseClientOptin, 480

Brightmail

- architecture, 476

- Configuration File options, 479-480

- deployment, 479

- requirements and performance, 478-479

bulk mail, 650-651

C

CA certificates, installing, 726-731

cacheeverything channel keyword, 371

cachefailures channel keyword, 371

cachesuccesses channel keyword, 371

cannot open alias include file, MTA error
messages, 860

caption, 425

cert8.db, 175

certificate-based login, 141-142, 733-734

certificates

- installing, trusted CA, 726-731

- obtaining, 722-731

certmap.conf, 734

certurl, 765

changing your configuration, 847

channel block, 200

channel-by-channel size limits, 408

channel host table, 236

channel/host table, 200

channel1, 236

channel processing, simultaneous requests, 257

channel programs, troubleshooting, 839

channel protocol selection, 362-363

channels

- alternates, 374

- channel-specific rule checks, 310

- character set labeling, 366

- comment lines in definitions, 200

- configuring, 317, 427

- connection caching, 371

Channels, creating, 850-852

channels

- defaults, setting, 318

- definitions, 200

- description, 193, 197

- directionality, 382

- eight-bit data, 367

- IDENT lookups, 372

- interpreting names of, 310

- job processing pools, 384

- keywords for, 361

- master programs, 197

- message queues, 199

- channels (*Continued*)
 nameserver lookups, 373
 predefined, 427
 protocol selection and line terminators, 363
 protocol streaming, 368
 reverse DNS lookups, 371
 SASL support, 376
 slave programs, 197
 SMTP authentication, 376
 SMTP option files, 253
 structure of, 200
 submit only, 419
 target host choice, 375
 TCP/IP MX record support, 373
 TCP/IP port selection, 371
 TLS keywords, 379
character set labeling, 366-368
CHARSET-CONVERSION, 406
charset7 channel keyword, 366
charset8 channel keyword, 366
charsetesc channel keyword, 366
checkehlo, 363
checkehlo channel keyword, 363
checkoverssl, 765
chunkingclient, 378
chunkingserver, 378
ciphers, about, 732-733
ClamAV, 499-504
Cloudmark, 509
cluster agent, 94
`comm_dssetup.pl`, 64-65
`comm_dssetup.pl` worksheet, 993
commadmin domain delete, 123
commadmin domain purge, 123
commadmin user delete, 123
command-line utilities
 mboxutil, 638
 MTA, 263
 reconstruct, 641
 stored, 642-643
commands, 672
COMMENT_STRINGS mapping table, 397
commentinc, 396
commentomit, 396
comments, in address message headers, 396-397
commentstrip, 396
commenttotal, 396
Communications Express, troubleshooting, 672
Communications Express Mail, 751
Communications Services, documentation, 41
compiled configuration version mismatch, 863
Compiling, MTA Configuration, 233-235
compliance archiving, 687
components, configure, 65-70
config files, 80-81, 626, 635
configuration
 components, 65-70
 high availability, 97-114
 initial runtime, 65-71
 optional flags, 65-70
 passwords, 121
 port numbers, 82-83
 Veritas Cluster Server, 116
configuration files
 aliases, 252
 blank lines in, 236
 conversion, 253
 Dispatcher, 253
 `imta.cnf`
 structure, 235
 Job Controller, 256
 mapping, 254
 MTA, 235
 `nsswitch.conf`, 374
 options, 255
 `sslpassword.conf`, 723-724
 tailor, 255
configurations files, dispatcher.cnf, 818-820
configuring SMTP blocking, 74-76
configutil
 `alarm.diskavail`, 889
 `alarm.msgalarmnoticehost`, 889
 `alarm.msgalarmnoticeport`, 889
 `alarm.msgalarmnoticercpt`, 889
 `alarm.msgalarmnoticesender`, 889
 `alarm.serverresponse`, 889
 `gen.newuserforms`, 131
 `gen.sitelanguage`, 134

configutil (Continued)

local.service.pab, 134
local.sso, 168
local.store.notifyplugin, 913
local.store.pin, 595-596
local.ugldapbasedn, 135
local.ugldapbinddn, 134, 135
local.ugldaphost, 134, 135
local.ugldapport, 135
local.ugdapuselocal, 135
local.webmail.sso, 168
logfile.service, 828
sasl.default, 718
sasl.default.ldap, 718
service.http, 153
service.http.plaintextmincipher, 146-150
service imap, 146-150
service imap.banner, 139
service.loginseparator, 140
service.pop, 145-146
service.pop.banner, 139
service.service, 745
store.admins, 594-595
store.defaultmailboxquota, 620
store.partition, 636
store.quotaenforcement, 623
store.quotaexceedmsginterval, 622
store.quotagraceperiod, 624
store.quotanotification, 621-623
store.quotawarn, 622
conflicts, port numbers, 82-83
conn_throttle, 579-586
conn_throttle.so, 555
connectalias, 393
connectcanonical, 393
connection caching, 371
connections, simultaneous, 937
Console, 124
content-transfer-encoding, 410
controlling error messages associated with
 rewriting, 313
conversion channel, 431
 alternative, 421
 bouncing messages, 444-446

conversion channel (Continued)

 configuration of, 432, 434
 conversion control, 253
 deleting messages, 444-446
 example, 446-450
 header management, 443
 holding messages, 444-446
 information flow, 437
 mapping table, 443-444
 output options, 441-443
 passing directives, 441-443
 processing, 435-444
 traffic for conversion processing, 434-435
conversion control, 253
conversion file, 253
conversion tag, 441
conversion tags, 440
conversions file, 435
converting addresses, 264
copysendpost, 280
copywarnpost, 280
core files, troubleshooting the message store, 663
correcting incomplete addresses, 393-394
corresponding channel characteristics, 374
counterutil, 879, 886
 alarm statistics, 880-881
 db_lock, 878
 diskusage, 881
 output, 879-880
 POP, IMAP, HTTP, 881
 serverresponse, 882
counterutil -l, 879
CRAM-MD5, 717
crladir, 766
crlenable, 766
crlmappingurl, 766
crlurllogindn, 766
crlurlloginpw, 766
crlusepastnextupdate, 767
crontab, 129-131
CTE field, 410

D

daemon channel keyword, 375
 data files, 80-81
 database, 265-266
 general text, 250
 database, general, 567
 database log files, troubleshooting the message store, 663
 date conversion, 402-403
 date fields, 402
 date specification, day of week, 403
 datefour, 402
 dates, two-digit, 402
 datetwo, 402
 day of week, date specification, 403
 dayofweek, 403
 debug, 491, 497
 debugging, 418
 dispatcher, 818-820
 debugging tools
 channel_master.log-* files, 845
 imsimta cache -view, 852
 imsimta process, 838
 imsimta qm, 836, 873
 imsimta qm start and stop, 840
 imsimta run, 840
 imsimta test -rewrite, 837, 864
 log_message_id, 842
 mail.log_current, 842
 mail.log_current records, 845
 mapping tables, 841
 master_debug, 843
 message file, 845
 slave_debug, 843
 subdirs, 844
 TCP/IP network
 PING, TRACEROUTE, and NSLOOKUP, 849
 tcp_local_slave.log-* file, 845
 default datasize, 820
 default error messages, rewrite and channel matching failures, 313
 defaultmx channel keyword, 373
 defaultnameservers channel keyword, 373
 defaults channel, 318

defaults channel (*Continued*)

 in a configuration file, 201, 236
 DEFER_GROUP_PROCESSING, 224
 deferralrejectlimit, 425
 deferred, 381, 382
 deferred delivery dates, 393-394
 deferred message processing, 383
 defragment, 406
 Defragmentation Channel, 406-407
 defragmentation of message, 406-408
 delegated administration, 123, 735-736
 Delegated Administrator, 72-73
 Delegated Administrator for Messaging, 123
 delete domain, 123
 delete user, 123
 deleted, 630
 deletemessagehash, 405
 DeleteMsg parameter, 699-700
 deleting messages, 593-594
 Delivery Failure, 873
 DELIVERY_OPTIONS, 220, 534, 535
 delivery reports, *See* notification messages
 delivery retry frequency, 383-384
 delivery status notifications, *See* notification messages
 denial-of-service, MeterMaid, 579-586
 Denial-of-Service Technologies, 509-510
 dequeue_removeroute, 399
 description, 425
 destination address, 416
 destinationfilter, 419, 572
 destinationnosolicit, 424
 destinationsspamfilterXoptin, 420
 destinationsrs, 519
 destinationtype parameter, 695
 DIAGNOSTIC_CODE, 277
 DIGEST-MD5, 717
 direct LDAP
 See also MTA
 settings, 231-232
 direction-specific rewrites, 311
 directories
 for log files, 822-823
 message store, 589
 directory, 201

- directory layout, 80-81
Directory Lookups Over SSL, 749
Directory Server, 134
 configuration settings, 134-135
 requirement, 134
 user directory, 122, 134
 worksheet, 991
directory server replica, 71-72
dirsync, 205
disabledestinationspamfilterX, 420
disableetrn, 364
disablesourcespamfilterX, 420
Discarded Messages, 573-574
 holding, 573-574
disclaimer text, 431
disconnectbadauthlimit, 410
disconnectbadcommandlimit, 416
disconnectrecipientlimit, 416
disconnectrejectlimit, 416
disconnecttransactionlimit, 416
Disk Space, 869-872
disk space
 monitoring, 642
 quotas for, 616-624
 reducing, 643-646
Disk usage, 887
Dispatcher
 configuration file, 253
 controlling, 196
 debugging and log files, 818-820
 description, 195
 MAX_CONNS option, 195
 MIN_CONNS option, 195
 MIN_PROCS option, 195
 restarting, 196
 starting, 196
 stopping, 196
dispatcher, troubleshooting, 847
dispatcher.cnf file, 818-820
dispatcher configuration file, 253, 818-820
disposition_option.dat, 276
dispositionchannel, 418
DNS
 domain verification, 366
DNS (*Continued*)
 IDENTprotocol, 372
 MX records, 373
 reverse lookups, 371, 372
DNS, configuring, 65-70
DNS Lookups, 563-566
DNS problems, MTA troubleshooting, 865
dns_verify, 563
documentation, where to find Communications Services documentation, 41
domain
 database, 314
 DNS verification, 366
 literals, 301
 removing, 123
 specification in an address, 297
 stopping inbound processing, 841
DOMAIN_FAILURE, 208, 209
DOMAIN_MATCH_URL, 206
 enabling direct LDAP, 231-232
domain preferred language, 133
DOMAIN_UPLEVEL, 206, 211, 212
domainetrn, 364
domainetrn channel keyword, 364
domainUidSeparator, 211
domainvrify, 365
dropblank, 395
duplicate aliases found, MTA error messages, 860
duplicate host in channel table, MTA error messages, 860
duplicate mapping name found, MTA error messages, 860

E

- EHLO, 360
ehlo, 363
ehlo channel keyword, 363
EHLO command, 363
Eight-Bit Data, 367-368
eight-bit data, 367
eightbit channel keyword, 367
eightnegotiate channel keyword, 367
eightstrict channel keyword, 368

-
- Encoded messages, 856-857
 encoded received message, 857
 encoding, 409
 encoding header, 402
 encryption, accelerators for, 723
 encryption settings, 135, 788-789
ENS
 administering, 913
 configuration parameters, 913-914
 configuring IMAP IDLE, 148
 configuring with JMQ notification plug-in, 689-690
 enable, 912
 sample programs, 912-913
 starting and stopping, 913
ENS_ACCESS, environment variable, 148
envelope To, address, 311
envelope to Address in Received, header, 396
environment variable, ENS_ACCESS, 148
error initializing ch_facility
 compiled character set version mismatch, 861
 no room in, 861
error messages
 cannot open alias include file, 860
 error initializing ch_facility, 861
 MTA, 859
 bad equivalence for alias, 860
 duplicate aliases found, 860
 duplicate host in channel table, 860
 duplicate mapping name found, 860
 local host too long, 861
 mapping name is too long, 861
 no equivalence addresses, 861
 no official host name for channel, 862
 official host name is too long, 862
 error notification messages, localizing, 271
 errors in mm_init, 859
 errsendpost, 280
 errwarnpost, 280
 /etc/nsswitch.conf, 848
 ETRN command, 364
 ETRN Command Support, 364-365
 Event Notification Service, 911-914
 See also ENS
 examples files, 80-81
 exclamation point (\!), 299
 exclusive, 629
 expandchannel, 388
 expandchannel channel keyword, 382
 expandlimit, 388
 expandlimit channel keyword, 382
 expansion of multiple addresses, 388-389
 expire, 625-635
 expire_exclude_list, 626, 635
 explicit routing, 392
 explicit routing, disable, 392-393
 expnallow, 365
 expndefault, 365
 expndisable, 365
 exproute, 391
 EXPROUTE_FORWARD option, 392
 expunge, 594
 ExpungeHeaders parameter, 699
 expunging messages, 593-594
 external modules (PKCS #11), 722-723

F

- failed delivery, 383-384
 failed delivery attempts, 280
 failed messages, 280
 failure of rewrite rules, 301
 field, 491, 497
 file, including in configuration files, 236
 file descriptors, 671-672
 file layout, 80-81
 file open or create errors, 864
 fileinto, 419
 files, header options, 402
 filesperjob, 384
 filesperjob channel keyword, 381
 filter, 419
 FILTER_DISCARD Channel, 573-574
 FILTER_JETTISON, 574
 filters, 541, 570
 See also mail filtering
 channel level, 570
 debugging user-level, 574-578
 IP Address, 555-556, 579-586

f
filters (*Continued*)

Messenger Express, 79
 MTA-wide, 570, 573
 per-user, 570
 Sieve, 224
 Sieve Extensions, 504-505
 folder, group/shared, 596-598
 folderpattern, 629
 folders, valid characters, 592-593
 foldersize, 629
 Forged Email Prevention, 511-520
 FORWARD address mapping, 268-271
 forward database, 265-266
 Forward Database, 268-271
 forwardcheckdelete channel keyword, 371
 forwardchecknone channel keyword, 371
 forwardchecktag channel keyword, 371
 forwarded messages, vacation, 539-540
 four-digit dates, 402
 fragmentation, of long messages, 408-409
 From\\, address, 392
 FROM_ACCESS mapping table, 543, 550
 fully qualified domain name (FQDN), 298

G

gen.newuserforms, 131
 gen.sitelanguage, 134
 general database, 265-266, 307
 general MTA error messages, 859
 general text database, 250, 566, 567
 generatemessagehash, 405
 generating character set labels, 366
 getent, 65-70
 greeting message, 131
 Per-Domain, 131-133
 group Expansion Attributes, 224-227
 group folder, 596-598
 groups, creation, 122
 groups, theory of operations, 224

H

hardware space, troubleshooting the message store, 661
 hashdir, 640-641
 HAStoragePlus, 96
 header
 handling keywords, 400-405
 language, 405
 maximum length, 404
 removing, 401-402
 Return-path, 396
 splitting long lines, 403
 stripping illegal blank recipient, 395
 X-Envelope-to, 402
 header_733, 390-391
 header_822, 390
 header alignment, 403-404
 HEADER_LIMIT, 415
 header options files, 402
 header trimming, 401
 header_uucp, 391
 headerlabelalign, 403
 headerlimit, 415
 headerlinelength, 403
 headerread, 401
 headerread keyword, 402
 headers, definitions, 432-434
 headertrim, 401
 heap size, 820
 HELD message queue files, 854-856
 .HELD messages, 854-856
 HIDE_VERIFY, 365
 high availability, 87
 additional configuration notes, 114
 High Availability, auto restart, 129
 high availability
 binding IP address, 114-115
 cluster agent, 94
 models, 87
 Sun Cluster, 97-114
 Sun Cluster prerequisites, 96
 unconfiguring, 119
 useconfig, 95
 hold channel, 431

-
- holdexquota, 414
 holdlimit, 388
 holdlimit channel keyword, 382
 host, 491, 497
 host/domain specifications, 297
 host location-specific rewrites, 312
 host name, extracting, 298
 hosts file, 65-70
 how to manually run a channel program, 839
 http logging, disable, 828
 HTTP message access, *See* message access
 HTTP service
 access control filters, 745
 certificate-based login, 141-142
 client access control, 145
 configuring, 151-155
 connections per process, 143
 disabling, 153
 dropping idle connections, 144
 enabling, 153
 logging out clients, 145
 login requirements, 140-142
 message settings, 151-155
 MTA settings, 151-155
 number of processes, 142
 password-based login, 153
 performance parameters, 142-145
 port numbers, 138-139
 proxy authentication, 745-746
 security, 714-715
 session ID, 715
 specialized web server, 151-155
 SSL port, 139
 starting and stopping, 124-127
 threads per process, 144
- |
- ibiff plug-in, and JMQ notification plug-in, 690
 iBiffconfiguration parameters, 913-914
 ICAP, 464
 option file, 497
 iddenttcpsymbolic channel keyword, 372
 IDENT lookups, 372
 identd, 742
 identify channels in message path, how to, 842
 identnone channel keyword, 373
 identnonelimited channel keyword, 373
 identnonenumERIC channel keyword, 373
 identnonesymbolic channel keyword, 373
 identTCP channel keyword, 372
 identTCPlimited channel keyword, 372
 identTCPnumeric channel keyword, 372
 IDLE (IMAP), configuring IMAP IDLE, 148-150
 idle connections, dropping, 144
 ignoremessageencoding, 410
 ignoremultipartencoding, 410
 ignoreencoding, 406
 iii_res* functions, slow SMTP server, 848
 illegal host/domain errors, 864
 MX record lookups, 864
 IMAP, *See* message access
 IMAP Access, limiting, 744
 IMAP FETCH, with message-type flags, 611
 IMAP SEARCH, with message-type flags, 611
 IMAP service
 access control filters, 745
 banner, 139, 146-150
 certificate-based login, 141-142, 733-734
 client access control, 145
 client debug, 661-662
 configuring, 146-150
 connection settings, 146-150
 connections per process, 143
 disabling, 146-150
 dropping idle connections, 144
 enabling, 146-150
 events slow, 675
 IMAP IDLE, configuring, 148-150
 login requirements, 140-142
 message types, 610-611
 monitoring user access, 659
 number of processes, 142
 password-based login, 719-720
 password-based long, 146-150
 performance parameters, 142-145
 port numbers, 138-139, 139
 process settings, 146-150

- IMAP service (*Continued*)
 readership utility, 641
 shared folders, 641
 SSL, 139, 721
 SSL port, 139
 starting and stopping, 124-127
 threads per process, 144
imesrestore, 653
imexpire
 See automatic message removal
 localized filepatterns, 630-631
 theory of operation, 625-626
immonurgent, 382
immonitor-access, 878
implicit routing, 392
improute, 391
IMPROUTE_FORWARD, 392
imquotacheck, 589, 623, 641, 886
ims50, 212, 215
imsbackup utility, 649
imsched, 129, 625-626, 633
imsconnutil, 659
imsimta cache -view, 852
imsimta counters, 882
imsimta crdb, 265-266
imsimta ims, 558
imsimta process, 838
imsimta qm, 836, 873
imsimta qm, 431
imsimta qm counters, 885
imsimta qm stop and start, 840
imsimta reload, 233
imsimta run, 840
imsimta test -exp, 574-576, 576, 577
imsimta test -rewrite, 574, 837, 864
 MTA troubleshooting, 836
imsimta test -rewrite -filter, 574
imsrestore utility, 649, 650
imta.cnf, 209, 235
imta.cnf configuration file, structure, 235
IMTA_LANG, 272
IMTA_MAPPING_FILE option, 237
IMTA_QUEUE, 200
INCLUDE_CONVERSIONTAG, 441
include files, 80-81
includefinal, 279, 283
incoming connection, 374
incoming mail, 847
incorrect handling of notification messages, looping
 messages, 853
inetCanonicalDomainName, 210
inetDomainStatus, 211
initial runtime configuration, 65-71
 silent, 70-71
inner, 401
inner header, rewriting, 395-396
inner header rewriting, 395-396
innertrim, 401
install files, 80-81
installer, silent, 70-71
installing messaging server and directory server
 replica, 71-72
INTERFACE_ADDRESS, 371
interfaceaddress channel keyword, 371
INTERNAL_IP mapping table, 74-76
internal modules (PKCS #11), 722-723
Internet Content Adaptation Protocol, 463
interpretencoding, 406
interpreting addresses, 391
interpretmessageencoding, 410
interpretmultipartencoding, 410
invalid address, 280
IP_ACCESS mapping table, 543
IP_ACCESS Mapping Table, 554-555
IP address, stopping inbound processing, 841
IP Address filtering, 555-556, 579-586
IP Address throttling, 579-586
IPv4 matching, 243
- J**
- jettison, 574
JMQ notification plug-in
 and Message Queue, 692-701
 configuring, 693-697
 configuring multiple plug-ins, 697-698
 default values of parameters, 704-705
 description, 689-692

JMQ notification plug-in (*Continued*)

- message properties, 705-712
- message types, 612
- notification messages, 701-703
- producing to a queue, 691
- properties carried with each message, 710-712
- publishing to a topic, 691
- specifying plug-in names, 696
- using multiple plug-ins, 691-692

jmqHost parameter, 694

jmqPort parameter, 695

jmqPwd parameter, 695

jmqQueue parameter, 695

jmqTopic parameter, 695

jmqUser parameter, 695

Job Controller

- commands, 257

job controller

- concepts, 202-204

Job Controller

- configuration file, 256

- examples of use, 257-261

- JOB_LIMIT option**, 258

- JOB_LIMIT pool option**, 202

- limits keywords, 385

- MAX_MESSAGES option**, 202, 203

- maxjobs channel option, 202

- restarting, 204

- SLAVE_COMMAND option**, 258

job controller

- start and stop, 204

Job Controller

- starting, 204

- stopping, 204

JOB_LIMIT, 385

JOB_LIMIT Job Controller option, 202, 258

junk email, removal, 625-635

K

keepmessagehash, 405

keywords

- table, 318-330, 330-359

L

language, 405

languages

- server site, 134

- user-preferred, 133

last resort host, 374

lastresort channel keyword, 374

LDAP MTA interface, 205

LDAP_ADD_HEADER, 227

LDAP_ADD_TAG, 227

LDAP_ALIAS_ADDRESSES, 217

LDAP_ATTR_DOMAIN1_SCHEMA2, 207

LDAP_ATTR_DOMAIN2_SCHEMA2, 207

LDAP_MAXIMUM_MESSAGE_SIZE, 226

LDAP_AUTH_DOMAIN, 225

LDAP_AUTH_PASSWORD, 226

LDAP_AUTH_POLICY, 225

LDAP_AUTH_URL, 226

LDAP_AUTOREPLY_ADDRESSES, 537

LDAP_AUTOREPLY_TEXT, 538

LDAP_CANT_DOMAIN, 225

LDAP_CANT_URL, 226

LDAP_CAPTURE, 217, 261-263

LDAP_CONVERSION_TAG, 220, 440

LDAP_DELIVERY_FILE, 219

LDAP_DELIVERY_OPTION, 220

LDAP directory

- configuring lookups in user directory, 134-135

- customizing lookups, 134

- MTA**, 201

- requirements, 134

- user directory, 122, 134

LDAP_DISK_QUOTA, 219

LDAP_DOMAIN_ATTR_ALIAS, 207

LDAP_DOMAIN_ATTR_AUTOREPLY_TIMEOUT, 211

LDAP_DOMAIN_ATTR_BASEDN, 207

LDAP_DOMAIN_ATTR_BLOCKLIMIT, 211, 219

LDAP_DOMAIN_ATTR_CANONICAL, 210

LDAP_DOMAIN_ATTR_CATCHALL_ADDRESS, 211, 213

LDAP_DOMAIN_ATTR_CATCHALL_MAPPING, 211

LDAP_DOMAIN_ATTR_CONVERSION_TAG, 211, 440

LDAP_DOMAIN_ATTR_DISK_QUOTA, 211

LDAP_DOMAIN_ATTR_FILTER, 211
LDAP_DOMAIN_ATTR_MAIL_STATUS, 211
LDAP_DOMAIN_ATTR_MESSAGE_QUOTA, 211
LDAP_DOMAIN_ATTR_OPTIN, 211
LDAP_domain_attr_optinX, 474
LDAP_DOMAIN_ATTR_RECIPIENTCUTOFF, 211, 415
LDAP_DOMAIN_ATTR_RECIPIENTLIMIT, 211, 415
LDAP_DOMAIN_ATTR_REPORT_ADDRESS, 211
LDAP_DOMAIN_ATTR_ROUTING_HOSTS, 206
LDAP_DOMAIN_ATTR_SMARTHOST, 211, 213
LDAP_DOMAIN_ATTR_SOURCE_CONVERSION_TAG, 211, 415
LDAP_DOMAIN_ATTR_SOURCEBLOCKLIMIT, 211, 415
LDAP_DOMAIN_ATTR_STATUS, 211
LDAP_DOMAIN_ATTR_UID_SEPARATOR, 211
LDAP_DOMAIN_FILTER_SCHEMA1, 207
LDAP_DOMAIN_ROOT, 207
LDAP_END_DATE, 223
LDAP Errors, handling, 213
LDAP_ERRORS_TO, 227
LDAP_EXPANDABLE, 227
LDAP_GROUP_DN, 227
LDAP_GROUP_OBJECT_CLASSES, 215
LDAP_GROUP_RFC822, 227
LDAP_GROUP_URL1, 226
LDAP_GROUP_URL2, 226
LDAP_HOST_ALIAS_LIST, 206
LDAP_LOCAL_HOST, 206
LDAP_MAIL_REVERSES, 229
LDAP_MESSAGE_QUOTA, 219
LDAP_MODERATOR_URL, 226
LDAP_OPTIN, 223, 466
LDAP_optinX, 473, 474
LDAP_PERSONAL_NAME, 537
LDAP_PREFIX_TEXT, 227
LDAP_PRESENCE, 224
LDAP_PROGRAM_INFO, 219
LDAP Provisioning Tools, 74
LDAP_RECIPIENTCUTOFF, 415
LDAP_RECIPIENTLIMIT, 415
LDAP_REJECT_ACTION, 225
LDAP_REJECT_TEXT, 225
LDAP_REMOVE_HEADER, 227
LDAP_REPROCESS, 224
LDAP_SCHEMATAG, 212
LDAP Server Failover, 136
LDAP_SOURCE_CONVERSION_TAG, 440
LDAP_SOURCE_OPTINX, 473
LDAP_SOURCEBLOCKLIMIT, 411
LDAP_SPARE_1, 219
LDAP_SPARE_2, 219
LDAP_START_DATE, 223
LDAP_SUFFIX_TEXT, 227
LDAP_USE_ASYNC, 231
LDAP_USER_OBJECT_CLASSES, 215
LDAP_USER_ROOT, 207
Legato, 653-656
less than sign (<), 236
lib files, 80-81
libspamass.so, 482
line length reduction, 409
line length restrictions, 409
linelength, 409
linelimit, 411
link count, 644
Linux, default base directory for, 41
LMTP, 521
 back end stores, no MTA, 527-528, 529
 configuring, 525
 configuring the relays, 526-527
 delivery features, 522
 protocol, 530-532
local.autorestart, 127, 887
local.autorestart.timeout, 129, 887
local channel, options, 430-431
local.enablelastaccess, 659
local.ens.enable, 126
local host too long, MTA error messages, 861
local.hostname, 206
local.http.enableuserlist, 659
local imap.enableuserlist, 659
local imta.enable, 126
local imta.hostnamealiases, 206
local imta.mailaliases, 212
local imta.schematag, 212
Local Mail Transfer Protocol, See LMTP

local.mmp.enable, 126
local.probe.service.timeout, 887
local.probe.service.warningthreshold, 887
local.probe.warningthreshold, 887
local.queuedir, 888
local.sched.enable, 126
local.schedule.expire, 634
local.schedule.msprobe, 129, 888
local.schedule.taskname, 129
local.service.pab, 134
local.smsgateway.enable, 126
local.snmp.cachettl, 901
local.snmp.contextname, 902
local.snmp.directoryscan, 901
local.snmp.enable, 126, 900
local.snmp.enablecontextname, 902
local.snmp.servertimeout, 901
local.snmp.standalone, 900
local.sso, 168
local.store.checkdiskusage, 871
local.store.expire.loglevel, 634, 635
local.store.notifyplugin, 913
local.store.overquotastatus, 619, 624
local.store.quotaoverdraft, 619, 624
local.store.relinker.enabled, 646
local.store.relinker.maxage, 646
local.store.relinker.minsize, 646
local.store.relinker.purgecycle, 646
local.store.sharedfolders, 602
local.store.snapshotinterval, 667
local.store.snapshotpath, 666
local.ugldapbasedn, 135
local.ugldapbasedn configutil, 207
local.ugldapbinddn, 134, 135
local.ugdaphost, 134, 135, 136
local.ugdapport, 135
local.ugdapuselocal, 135, 136
local.watcher.enable, 127, 129, 888
local.webmail.cert.enable, 770
local.webmail.cert.port, 770
local.webmail.smime.enable, 771
local.webmail.sso, 168
local.webmail.sso.amcookiename, 159
local.webmail.sso.amloglevel, 159
local.webmail.sso.amnamingurl, 158
local.webmail.sso.id, 168
local.webmail.sso.prefix, 169
local.webmail.sso.singlesignoff, 159
localizing, notification messages, 271
localvrfy channel keyword, 365
location-specific rewrites, 311
LOG_CONNECTION, 799
LOG_CONNECTION option, 804
LOG_FILENAME, 799
LOG_FILENAME option, 804
log files, 80-81
 troubleshooting the message store, 661
 troubleshooting the MTA, 838
LOG_MESSAGE_ID, 799
log_message_id, 842
LOG_MESSAGE_ID option, 804
LOG_MESSAGES_SYSLOG option, 801
LOG_NOTARY, 799
LOG_PROCESS, 799
LOG_PROCESS option, 805
LOG_QUEUE_TIME option, 804
LOG_TRANSPORTINFO, 360
LOG_USERNAME option, 805
logfile.service, 828
logfile.service.loglevel, 828
logging, 417, 791
 analyzing logs, 795
 architecture of, 824
 categories, 822
 channels, 796
 directories for log files, 822-823
 enabling MTA, 800-801
 file format, 823-824
 files, 792
 levels of, 821-822
 LOG_CONNECTION option, 804
 LOG_FILENAME option, 804
 LOG_MESSAGE_ID option, 804
 LOG_MESSAGES_SYSLOG option, 801
 LOG_PROCESS option, 805
 LOG_QUEUE_TIME option, 804
 LOG_USERNAME option, 805
managing service logs, 820-833

logging (Continued)

 Message Store, 831-833
 message store and administration server, 820
 MTA, 796, 800
 MTA entry codes, 797
 MTA entry modifier codes, 798
 MTA Examples, 805-818
 MTA Message and Connection, 796-820
 Options, 824-826
 options, 825
 SEPARATE_CONNECTION_LOG option, 805
 severity levels, 821-822
 tools for Managing, 795
 types, 792
 viewing logs, 826-827

logheader, 417

login

- certificate-based, 141-142, 733-734
- password-based, 719-720

login separator, for POP, 140

login service, password-based login, 141

logindn, 767

loginpw, 767

long-term service failures, 280

loopcheck, 418

looping messages, 853

- incorrect handling of notification messages, 853
- postmaster address is broken, 853

M

MAIL_ACCESS mapping table, 543, 548

Mail conversion tag, 440

mail filtering

- channel-level filters, 570
- description, 541
- mapping tables, 542
- MTA-wide filters, 570
- per-user filters, 570
- server-side rules, 570

mail forwarding, 373

Mail Forwarding, SPF problems, 518-520

mail.log_current, 842

mailAllowedServiceAccess, 781

mailAlternateAddress, 212

mailAutoReplyMode, 538

mailAutoReplyText, 538

mailAutoReplyTextInternal, 539

mailAutoReplyTimeOut, 539

mailbox, protection, 595-596

mailbox encoding

- restricted, 395-396

mailbox names, valid characters, 592-593

mailbox specifications, 395

mailboxes

- automatic message removal, 625-635
- managing, 638-641
- mboxutil utility, 638
- migrating, 675-686
- reconstruct utility, 667-671
- repairing, 667-671

mailConversionTag, 220

mailDeferProcessing, 224

mailDeliveryOption, 220, 534

mailDomainCatchallAddress, 211

MailDomainConversionTag, 440

mailDomainConversionTag, 211

mailDomainDiskQuota, 618

mailDomainMsgMaxBlocks, 211

mailDomainMsgQuota, 619

mailDomainReportAddress, 211

mailDomainSieveRuleSource, 211

maildomainstatus, 624

mailDomainStatus, 211, 619

mailEquivalentAddress, 212

mailfromdnsverify channel keyword, 366

mailinglist, creation, 122

mailMessageStore, 637

mailMsgMaxBlocks, 219

mailMsgQuota, 618

mailQuota, 219, 618

mailRejectText, 225

mailRoutingAddress, 218

mailRoutingHosts, 206

mailRoutingSmartHost, 211

MailSieveRuleSource, 574

mailSieveRuleSource, 224

mailUserStatus, 618

mailuserstatus, 624
maintain data, 605-606
manually running a channel program, 839
mapping, / matching, 243
mapping entry patterns, 241-243
mapping entry templates, 244-251
mapping file, 237, 254
 file format, 239
 locating and loading, 237
mapping name is too long, MTA error messages, 861
mapping operations, 241
mapping pattern wildcards, 241-243
mapping probes, 247
mapping table
 COMMENT_STRINGS, 397
 NOTIFICATION_LANGUAGE, 272
mapping tables, 238, 841
 See also access control
 description, 542
 FROM_ACCESS, 543
 handling large numbers of entries, 566-568
 IP_ACCESS, 543
 list of them all, 238
 MAIL_ACCESS, 543
 ORIG_MAIL_ACCESS, 543
 ORIG_SEND_ACCESS, 543
 PORT_ACCESS, 543, 555
 SEND_ACCESS, 543
 SMS_Channel_TEXT, 930
 X-REWRITE-SMS-ADDRESS, 929
mapping template substitutions and
 metacharacters, 244-245
master, 382
master_command, 258
master_debug, 417, 843
master program, 257, 382
matching any address, 294
matching procedure, rewrite rules, 299
MAX_CLIENT_THREADS, 360
max_client_threads, 385
MAX_CONNS, 528
MAX_CONNS Dispatcher option, 195
MAX_HEADER_BLOCK_USE, 409
MAX_HEADER_LINE_USE, 409
MAX_LIFE_CONNS, 528
MAX_LIFE_TIME, 528
MAX_MESSAGES Job Controller option, 202, 203
MAX_PROCS, 528
MAX_PROCS Dispatcher option
 Dispatcher
 MAX_PROCS option, 195
MAX_PROCS*MAX_CONNS, 848
maxblocks, 408
maxBodySize parameter, 699
maxheaderaddrs, 403
maxheaderchars, 403
maxHeaderSize parameter, 699
maximum length header, 404
maxjobs, 384
maxjobs channel keyword, 202, 381
maxlines, 408
maxprocchars, 404
maysaslserver, 376
maytls, 733
maytls channel keyword, 379
maytlscclient channel keyword, 379
maytlsserver channel keyword, 379
mboxutil, 638-640
MD5, 644
MDN, 284
memberURL, 226
message
 automatic removal, 625-635
 dequeue, 393
 fragmentation, 411
 migration, 675-686
 purge, 625-635
 removal, 593-594
 size limits, 410-415
 without recipient header, 394
message access, 137
 general configuration, 137
 HTTP, 137-155
 HTTP Services, 137-155
 IMAP, 137-155
 log In w/o Domain Name, 140-141
 login requirements, 140-142
 password-based, 141

- message access (*Continued*)
POP, 137-155
POP, IMAP, or HTTP, 138
ports, encrypted, 139
service port numbers, 138-139
- message breakdown, 845
- message defragmentation, 406-408
- Message Disposition Notifications, 284, 533
- Message Disposition Notifications,
customizing/localizing, 284-285
- message disposition notification *See also*,
notifications, 276
- message expiration, 625-635
- Message-hash:, 405
- message header, date fields, 402
- message header lines, trimming, 402
- Message Queue
description, 689-690
designing a JMQ notification plug-in, 692-701
- message queue directories, troubleshooting, 836
- message queues, 199, 872
sizing disks, 199
- message queues, monitoring, 873
- message rejection, 411
- Message Store, 65-70
- message store
access control, 594-596
- Message Store
adding disk space, 638
- message store
administrator access, 594-596
aging policies, 625-635
archiving, 687
automatic message removal, 625-635
backup, excluding trash, 650-651
backup groups, 648
backup policies, 647
checking and repairing mailboxes, 669-670
command-line utilities, 588-589
common problems and solutions, 671-675
configuring disk quotas, 616-624
configuring partitions, 635-637
deleting messages, 593
directory layout, 589
- message store (*Continued*)
disk space reduction, 643-646
expunging messages, 594
grace period, 624
group folder, 596-598
imsbackup utility, 649
imsrestore utility, 650
incremental backup, 650
logging, 791, 820
logging examples, 831-833
maintenance and recovery procedures, 638-646
managing message types, 606-616
- Message Store
mboxlist Database Log file, 887
- message store
message tracing, 830-831
overview, 587-589
partition, 636
partition, changing default, 637
partitions, 624
primary partition, 635
purging messages, 594
quotas (see also, quotas), 620-624
RAID technology, 635
Rebuild Mailboxes, 668-669
reconstruct utility, 667
Remove Orphan Accounts, 640
restoring data, 650
shared folder, 596-598
stored utility, 642-643
troubleshooting, 660
using Legato Networker for backup, 653
using third party software, 656
- Message Transfer Agent., *See* MTA
- message types
administering quotas, 612-614
configuring, 608-610
defining in a message header, 608
expiring and purging, 614-616
in IMAP FETCH session, 611
in IMAP SEARCH session, 611
in unified messaging applications, 607-608
managing in the message store, 606-616
Message Queue notifications, 612

-
- message types (*Continued*)
 removing, 614-616
 telephone front-end system, 607-608
 using with IMAP commands, 610-611
- messagecount, 629
- messagedays, 629
- messages not delivered, 852
- messages not dequeued, 849
- messagesize, 629
- messagesizedays, 629
- Messaging Multiplexor
See also MMP
 certmap plugins, 175
 configuration, 179, 186
 description, 173
 DNComps, 175
 encryption, 175
 example topology, 183
 features, 173
 FilterComps, 175
 how it works, 173-174
 IMAP example, 184
 POP example, 185
 pre-authentication, 176
 pre-configuration, 179
 set up, 178-181
 SSL, to use with, 181
 starting/stopping/refresh, 181
 store administrator, 175
 vdmap, 177
 virtual domains, 176
- Messaging Server
 worksheet, 65, 994
- Messenger Express, 65-70, 137
 debug, 661-662
 monitoring user access, 659
 troubleshooting, 672
 unknown/invalid partition, 673
- Messenger Express Mail Filters, 79
- Messenger Express Multiplexor, 151-155, 171
- metacharacters in mapping templates, 244-245
- MeterMaid, 579-586
- mgmanMemberVisibility, 227
- mgrpAddHeader, 227
- mgrpAllowedBroadcaster, 226
- mgrpAllowedDomain, 225
- mgrpAuthPassword, 226
- mgrpBroadcasterPolicy, 225
- mgrpDeliverTo, 226
- mgrpDisallowedBroadcaster, 226
- mgrpDisallowedDomain, 225
- mgrpErrorsTo, 227
- mgrpModerator, 225, 226
- mgrpMsgMaxSize, 226
- mgrpMsgPrefixText, 227
- mgrpMsgRejectAction, 225
- mgrpMsgSuffixText, 227
- mgrpRemoveHeader, 227
- mgrpRFC822MailMember, 227
- Microsoft Exchange, 379
- migrating mailboxes, 675-686
- migrating users, 431
- migration, message store size, 643
- Milter, 506-509
 deploying, 508-509
- MIME
 Headers, 432-434
 message construction, 432
 overview, 432-434
 processing, 405-410
- MIN_CONNS Dispatcher option, 195
- MIN_PROCS Dispatcher option, 195
- MISSING_RECIPIENT_POLICY, 394
- missingrecipientpolicy, 394
- mm_debug, 843
 debugging tools
 mm_debug, 839
- mm_init, 859
- MMP, 65-70
- MMP, 746
 AService.cfg file, 180
 AService-def.cfg, 180
 ImapMMP.config, 180
 ImapProxyAService.cfg file, 180
 ImapProxyAService-def.cfg, 180
 LDAP Server failover, 187
 modifying an existing instance, 181
 PopProxyAService.cfg file, 180

- MMP (*Continued*)
 PopProxyAService-def.cfg, 180
 SMTP Proxy, 178
 SsmtpProxyAService.cfg, 180
 SsmtpProxyAService-def.cfg, 180
- MMP, *See also* Messaging Multiplexor
- MobileWay, 960-961
- mode, 492, 498
- modifying, 272
- modifying passwords, 121
- monitoring, 867
- automatic restart, 127
 - CPU usage, 872
 - delivery failure rate, 873
 - delivery times, 869
 - disk space, 869
 - dispatcher, 874-875
 - httpd, 875-876
 - imapd, 875-876
 - job controller, 874-875
 - LDAP Directory Server, 875
 - LDAP server, 878
 - log files, 868
 - message access, 875-876
 - message queues, 872-873
 - message store, 877-878
 - message store database locks, 878
 - msprobe, 868, 886-889
 - MTA, 872-875
 - POP and IMAP servers, 878
 - popd, 875-876
 - postmaster Mail, 868
 - SMTP connections, 873-874
 - stored, 877-878
 - system performance, 869-872
 - tools &, 878-889
 - user access, 659-660
 - watcher, 867, 886-889
 - Webmail services, 875
- move user mailbox, 646
- moving mailboxes, 636-637
- msexchange, 379
- msg-svr-base, 591
- msg_svr_base*, 80-81
- msgcert, 724-725
- MsgFlags parameter, 701
- msprobe, 127, 886-889
- MTA, 65-70
- MTA, 859
- adding relaying, 557-559
 - alias expansion, 209
 - architecture, 193
 - archiving messages, 687
 - channels, 193, 197
 - command-line utilities, 263
 - concepts, 189
 - Configuration File, 235-237
 - configuration files, 235, 251
 - data flow, 205
 - directory information, 201
 - Dispatcher, 195
 - error handling, 208
 - Error Messages, 859-865
 - imta.cnf rewrite rule, 209
 - LDAP interface, 205
 - logging, 791, 796
 - message flow, 193-194
 - message queues
- See also* message queues
- Problems and Solutions, 846-859
- relay blocking, 560
- rewrite rules, 196, 206
- server processes, 195-196
- setting global options, 255
- theory of operations, 205
- troubleshooting, 835
- MTA channels, starting and stopping, 840
- MTA configuration, troubleshooting, 836
- MTA configuration file, 235
- MTA error messages, 859
- bad equivalence for alias, 860
 - cannot open alias include file, 860
 - duplicate aliases found, 860
 - duplicate host in channel table, 860
 - duplicate mapping name found, 860
 - error initializing ch_facility
 - compiled character set version mismatch, 861
 - no room in, 861

MTA error messages (*Continued*)

- local host too long, 861
- mapping name is too long, 861
- no equivalence addresses, 861
- no official host name for channel, 862
- official host name is too long, 862

MTA example

- message breakdown, 845
- start and stop channels, 843

MTA functionality, 189

MTA mapping file, 237

MTA-Only, 126-127

MTA optimizations, 285-287

MTA queues, 872

MTA troubleshooting, network and DNS problems, 865

MTA troubleshooting example, 842

multiple, 415

multiple \$M clauses, 310

multiple addresses, 415-416

multiple destination addresses, 415

multiple outgoing channels, 374

Multiplexor., *See* Messaging Multiplexor.

mustsaslserver, 376

musttls, 733

musttls channel keyword, 379

musttlscclient channel keyword, 379

musttlsserver channel keyword, 379

mx channel keyword, 373

MX record lookups, 864

MX record support, 373

myprocmail, with the Pipe channel, 429

N

nameparameterlengthlimit, 414

nameserver lookups, 373

nameservers channel keyword, 373

NDAAAuth-applicationID, 168

Net-SNMP, 894-902

netstat, 874

Network Appliance Filers, 638

network problem, 873

network services, 257

NewMsg parameter, 698-699

NIS, 65-70

nms41, 212, 215

no equivalence addresses, MTA error messages, 861

no official host name for channel, MTA error messages, 862

noaddressrs, 519

noaddreturnpath, 396

nobangoverpercent, 391

nobangoverpercent keyword, 298

noblocklimit, 411

nocache channel keyword, 371

nochunkingclient, 378

nochunkingserver, 378

nodayofweek, 403

nodeferred, 381, 382

nodefragment, 406

nodeestinationfilter, 419

nodestinationsrs, 519

nodropblank, 395

noehlo, 363

noehlo channel keyword, 363

noexproute, 391

noexquota, 414

nofileinto, 419

nofilter, 419

noheaderread, 401

noheadertrim, 401

noimproute, 391

noinner, 401

noinnertrim, 401

nolinelimit, 411

nologging, 417

noloopcheck, 418

nomailfromdnsverify channel keyword, 366

nomaster_debug, 417

nomsexchange, 379

nomx channel keyword, 373

non-ASCII characters, 859

non-delivery reports, *See* notification messages

noneInbox parameter, 703

nonrandommx channel keyword, 373

nonstandard message formats, converting, 406

nonurgentbackoff channel keyword, 381, 383

nonurgentblocklimit, 387
nonurgentblocklimit channel keyword, 381
nonurgentnotices, 278
nonurgentnotices channel keyword, 382
noreceivedfor, 396
noreceivedfrom, 396
noremotehost, 393
noretturnpersonal, 281
noreverse, 266, 395
normalbackoff, 383
normalbackoff channel keyword, 381
normalblocklimit, 387
normalblocklimit channel keyword, 381
normalnotices, 278
normalnotices channel keyword, 382
norules, 399
norules channel keyword, 310
nosasl, 376
nosaslserver, 376
nosaslswitchchannel, 376
nosendetrn, 364
nosendpost, 280
noservice, 389
noslave_debug, 417
nosmtp channel keyword, 363
nosourcefilter, 419
nosourcesrs, 519
noswitchchannel keyword, 374
notaries, 271
notary, *See* notification messages
notices, 278, 383
notices channel keyword, 382
NOTIFICATION_LANGUAGE mapping table, 272, 274
notification message, 279
Notification Messages, 282-283
notification messages
 additional features, 277
 blocking content return, 278
 constructing &, 272
 customizing and localizing, 274-276
 default values, 704-705
 internationalizing, 276-277

notification messages (*Continued*)
 removing non-US-ASCII characters from
 headers, 278
 sending/blocking to postmaster, 280
 setting delivery intervals for undeliverable
 mail, 278-279
notificationchannel, 418
notls channel keyword, 379
notlsclient channel keyword, 379
notlsserver channel keyword, 379
novrfy, 364
nowarnpost, 280
nox_env_to, 402
nsswitch.conf, 65-70
nsswitch.conf file, 374

O

official host name is too long, MTA error
 messages, 862
optin_user_carryover, 475
optional flags, 65-70
options, SLAVE_COMMAND, 261
options file, 255
OR_CLAUSES, 225
ORCPT, 547
ORIG_MAIL_ACCESS mapping table, 543, 548
ORIG_SEND_ACCESS mapping table, 543, 546
ORIGINAL_ADDRESS, 277
original recipient, 547
orphaned accounts, 640
os_smtp_* errors, 865
os_smtp_open errors, 865
os_smtp_read errors, 865
os_smtp_write errors, 865
ownership of files, troubleshooting, 837

P

parameterlengthlimit, 414
partial messages, 406-408
partition, invalid, 673

-
- partitions
 configuring for message store, 635-637
 full, 636-637
 message store, 624
 moving mailboxes between, 636-637
 primary, 635
 RAID technology, 635
- password authentication
 See also login
 HTTP service, 141
 IMAP service, 141
 POP service, 141
 SMTP service, 720
- password file (for SSL), 723-724
- password login, 719-720
- passwords, 121
- PDU, 925
- percent hack, 298
- percent hack rules, 293
- percent sign (%), 310, 313
- percentonly, 391
- percents, 390
- performance, relays, 521
- Performance and Tuning, 79-80
- performance enhancement, LMTP, 521
- performance parameters
 connections per process, 143
 number of processes, 142
 threads per process, 144
- periodic message return job, 281
- Persistent parameter, 696
- personal names in address message headers, 397-398
- personalinc, 397
- personalomit, 397
- personalstrip, 397
- pipe channel, 419, 429
- PKCS #11, internal and external modules, 722-723
- platformwin, 767
- pool, 384
- pool channel keyword, 381
- POP, *See* message access
- POP Before SMTP, 746-749
- POP over SSL, 145-146
- POP service
 access control filters, 745
 banner, 139
 certificate-based login, 733-734
 client access control, 145
 client debug, 661-662
 configuring, 145-146
 connections per process, 143
 dropping idle connections, 144
 login requirements, 140-142
 monitoring user access, 659
 number of processes, 142
 password-based login, 719-720
 performance parameters, 142-145
 port numbers, 138-139
 SSL, 721
 starting and stopping, 124-127
 threads per process, 144
- PORT, 371
- port, 492, 498
- PORT_ACCESS, 528, 552
- PORT_ACCESS mapping table, 543, 555
- PORT_ACCESS Mapping Table, 552-554
- port channel keyword, 371
- port numbers, 82-83
- post-install directory layout, 80-81
- post-install port numbers, 82-83
- post-installation configuration
 configuration
 SMTP blocking, 74-76
 port numbers, 82-83
 startup across reboots, 76
- postheadbody, 281
- postheadbody channel keyword, 283
- postheadonly, 281
- postheadonly channel keyword, 283
- postmaster, addresses, 281-283
- pre-authentication (Messaging Multiplexor), 176
- preferred language, domain, 133
- preferredLanguage, 133
- preparing LDAP Directory for Messaging
 Server, 64-65
- Priority parameter, 696
- processes, number of, 142

processing messages, 431
program, sending a message to, 431
program delivery
 pipe channel, 429
 setting up, 429
programs
 master, 257
 slave, 257
protocol streaming, 368
provisioning, 72-73
provisioning options, LDAP Provisioning Tools, 74
publish-and-subscribe, 911
purge, 594

Q

Q records, 873
queues, 872
queues, message, 199
quota checking report, 886
quotas
 attributes, 618-619
 configuring, 616-624
 configutil parameters, 619
 default, 620
 disabling, 623
 disk space, 616-624
 Domain, 621
 domain, 623
 enabling enforcement, 623
 Enforcement, 623-624
 enforcement, 623
 family groups, 623
 for message types, 612-614
 grace period, 624
 message, 616-617
 Netscape Messaging Server, 624
 notification, 621-623, 623
 usage, 641
 user, 616-617
 users, 621
 warnings, 621-623
quoted local-parts, 395
quoted-printable, 410

R

RAID technology, for message store, 635
randommx channel keyword, 373
RBL Checking, 563-566
readership, 601, 641
readsigcert, 767
received message, encoded, 857
receivedfor, 396
receivedfrom, 396
RECIPIENT_ADDRESS, 277
recipientcutoff, 414
recipientlimit, 414
recompile, MTA, 233, 251
reconstruct, 667, 669
 performance, 670-671
reconstruct command-line utility, 641
record keeping archiving, 687
recovery tasks
 mailboxes, 667-671
 reconstruct utility, 641
regulatory archiving, 687
rejectsmplonglines, 414
relay blocking, 560
relay blocking, removal of, 557-559
relaying, adding, 557-559
relaying mail, 873
relinker, 643, 644
 command line mode, 644
 Realtime Mode, 645-646
 Theory of Operations, 643-644
reload, 233
remote system, 374
remotehost, 393
remove domain, 123
remove user, 123
removing messages, 593-594
repeated percent signs, 299
replica, 71-72
requirements, Sun Cluster, 96
resolv.conf, 65-70
resource.properties, 168
restoring, using Legato Networker, 656
restoring incremental back-ups, 652-653
restoring the message store, 646

-
- restoring the message store, considerations, 651-653
restricted, 395
restricted channel keyword, 395
restricted mailbox encoding, 395-396
restrictions, line length, 409
return_option.dat, 276
RETURN_PERSONAL, 277
returnaddress, 281
returned message, content, 281
returnenvelope, 280, 283
returnpersonal, 281
Reversal Cache, 217
reverse, 395
REVERSE_ADDRESS_CACHE_SIZE, 230
reverse channel keyword, 267
reverse database, 264, 265-266
 channel-specific, 395
REVERSE_ENVELOPE, 266
reverse mapping, 264
Reverse Mapping, 267
REVERSE mapping table, 264
REVERSE mapping table flags, 265
REVERSE_URL, 229
 enabling direct LDAP, 231-232
revocationunknown, 768
rewrite, inner header, 395-396
rewrite process failure, 297
rewrite rules, 206, 236
 bang-style, 293
 blank lines, 236
 case sensitivity in templates, 296
 checks, 399
 control sequences, 302-313
 description, 196
 direction-specific, 311
 domain literals, 301
 example, 315-316
 failure, 301
 Finishing the Rewriting Process, 300-301
 handling large numbers, 314
 host location-specific, 312
 location-specific, 311
 match any address, 294
 operation, 297-302
 rewrite rules (*Continued*)
 ordinary templates A%B@C, 295
 pattern matching, 297
 patterns and tags, 291
 percent hacks, 293
 repeated templates A%B, 295
 scanning, 299-300
 specified route templates A@B@C, 296
 structure, 290
 substitution, username and subaddress, 305
 substitutions, customer-supplied routine, 308-309
 substitutions, general database, 307-308
 substitutions, host/domain and IP Literal, 305-306
 substitutions, LDAP Query URL, 306-307
 substitutions, literal character, 306
 substitutions, single field, 309-310
 substitutions, specified mapping, 308
 syntax checks after rewriting, 301
 tagged rule sets, 294
 template substitutions, 302-313
 templates, 294-296, 300
 testing, 314
 UUCP addresses, 293
 rewriting, inner header, 395-396
 rewriting an address, extracting the first host/domain specification, 297
 rewriting error messages, 313
RFC 2476, 419
RFC 2741, 895-896
RFC 3507, 463
rfc822MailMember, 227
ROUTE_TO_ROUTING_HOST, 206
routelocal, 392
routing
 explicit, 392
 implicit, 392
Routing Address, 218
routing information in addresses, 391-392
rules, 399
rules channel keyword, 310
runtime configuration, 65-71

S

S/MIME, 751
 Applet, 756-758
 Basic Configuration, 758-762
 conceptual prerequisites, 752
 defined, 751-752
 downloading applet, 757-758
 Getting Started, 756-763
 key pairs, 755
 LDAP credentials, 763
 LDAP directory, 762
 LDAP password pairs, 763
 Multi-language Support, 756
 options, 770
 private and public keys, 754
 Public Keys in LDAP Directory, 755
 required software/hardware, 753-754
 Smart Card, 754
 smart cards, 755
 smime.conf File, 764-770
 SSL, 771-772
 User Permissions, 756
s private/public key, 775-776
SASL
 channel keywords, 376
 description, 715
 sasl.default.auto_transition, 716, 718
 sasl.default.ldap, 718
 sasl.default.ldap.has_plain_passwords, 716
 sasl.default.ldap.searchfilter, 717
 sasl.default.ldap.searchfordomain, 717
 sasl.default.mech_list, 717, 719
 sasl.default.transition_criteria, 716
 saslswitchchannel, 374, 376
SASVE
 configuration example, 495-497
 deploying, 494-495
savedays, 629
SAVSE
 deploying, 494
 Options, 497-499
 overview, 494
 Requirements and Usage Considerations, 494
sbin files, 80-81

scheduling tasks, 129-131
Secure/Multipurpose Internet Mail Extension, *See* S/MIME
security
 about, 713-714
 authentication mechanisms, 715
 certificate-based login, 141, 733-734
 client access controls, 145
 client access to TCP services, 737-746
 concerns, 136
 HTTP service, 145, 714-715
 IMAP service, 145
 password-based login, 141
 POP service, 145
S/MIME
 See S/MIME
SASL, 715
SMTP service, 720
SSL, 721
TLS, 720
seen, 630
SEND_ACCESS mapping table, 543, 546
sendencryptcert, 768
sendencryptcertrevoked, 768
Sender Policy Framework, 511-520
Sender Rewriting Scheme, 518-520
sendetrn, 364
sendmail, clients, 77-78
sendpost, 280
sendsigncertrevoked, 768
sensitivitycompanyconfidential, 404
sensitivitynormal, 404
sensitivitypersonal, 404
sensitivityprivate, 404
SEPARATE_CONNECTION_LOG option, 805
separator, setting, 140
Server Response Time., 886
server-side rules, 570
Server-Side Rules, not working, 857-858
server-side rules, troubleshooting, 857
service, 389
service.{imap|pop|http}.plaintextmincipher, 717
service banners, 139
Service Conversions, 389

service.defaultdomain, 211
 service denial attack, 874
 service.http, 153
 service.http.enable, 126, 828
 service.http.enablesslport, 153, 828
 service.http.idletimeout, 154
 service.http.maxmessagesize, 154
 service.http.maxsessions, 154
 service.http.maxthreads, 154
 service.http.numprocesses, 154
 service.httpplaintextmincipher, 146-150, 153
 service.http.port, 153
 service.http.sessiontimeout, 154
 service.http.smtphost, 154
 service.http.smtpport, 155
 service.http.spooldir, 154
 service.http.sslport, 153
 service imap, 146-150
 service imap.allowanonymouslogin, 716
 service imap.banner, 139, 146-150
 service imap.enable, 126
 service imap.enablesslport, 146-150
 service imap.idletimeout, 146-150
 service imap.maxthreads, 146-150
 service imap.numprocesses, 146-150
 service imap.port, 146-150
 service imap.sslport, 146-150
 service loginseparator, 140
 service.pop, 145-146
 service.pop.banner, 139, 145-146
 service.pop.enable, 126, 145-146
 service.pop.enablesslport, 145-146
 service.pop.idletimeout, 145-146
 service.pop.maxsessions, 145-146
 service.pop.maxthreads, 145-146
 service.pop.numprocesses, 145-146
 service.pop.sslport, 145-146
 service.readtimeout, 888
 services
 enabling and disabling, 138
 HTTP, 137
 IMAP, 137
 MTA, 189, 233
 POP, 137

services (*Continued*)
 SMTP, 189, 233
 starting and stopping, 124-127
 sevenbit channel keyword, 367
 severity levels (of logging), 821-822
 shared folders, 596-598
 Access Control Rights, 601-602
 ACLs, 601-602
 distributed, 598, 603-604
 enable or disable, 602
 monitor &, 605-606
 public folders, 599-600
 shared folders, IMAP, 641
 Short Message Service, defined, 915
 Sieve, 574
 sieve, 630
 Sieve
 See also filters, user-level
 Sieve filtering language, 568
 silent installation, 70-71
 silenttetrn, 364
 silenttetrn channel keyword, 364
 sims40, 215
 sims401, 212
 simultaneous connections, controlling, 937
 single, 375, 415
 single channel keyword, 376
 single destination system per message copy, 415
 single sign-on
 See SSO
 Messenger Express configuration parameters, 167
 single_sys, 256, 375, 415
 single_sys channel keyword, 376
 slapd, 875
 slapd Problems, 875
 slave, 382
 SLAVE_COMMAND Job Controller option, 258
 SLAVE_COMMAND option, 261
 slave_debug, 417, 843
 slave program, 257, 382
 smart cards, 755
 SMIME
 CA Certificates in LDAP, 782-783
 certificate revocation, 780-781

SMIME (*Continued*)

Communications Express S/MIME End User Information, 786-789
CRL access, 777-778
CRL access trouble, 780
CRL checking, 776
CRL checking and proxy servers, 778
Enabling the Java Console, 789
finding user', 775-776
logging in, first time, 787-788
managing certificates, 782-786
Message Time, 779-780
Network Security Services (NSS, 786
permissions, 781-782
public keys and certificates in LDAP, 783-784
signature &, 788-789
Stale CRL, 778-779
verifying key/certificates in LDAP, 784-786
verifying Private and Public Keys, 774-781

SMPP V3.4, 925**SMS, 915**

adding more channels, 958-959
address validity checks, 929-930
Channel Definition and Rewrite Rules, 935-937
channel option file, 937-938
channel options, 938
configuration, 935
converting email to SMS, 919-925
debugging, 961
delivery retries, 959-960
email conversion options, 941-945
formatting templates, 957-958
Localization Options, 953-957
site-defined text conversions, 930-934
SMS options, 945-951

SMS channel, 915

attributes, 918
operation, 918
requirements, 917

SMS channel, adding, 935-937**SMS Channel, sample configuration, 960-961****SMS_Channel_TEXT mapping table, 930****SMTP AUTH, 557****SMTP Authentication, 746****SMTP Banner Delay, 510****SMTP blocking, post-installation configuration, 74-76****SMTP Channel, 359-380****smtp channel keyword, 363****SMTP channel option file, 747****SMTP Channel Threads, 387-388****SMTP chunking, 378-379****smtp_client process, 523****SMTP Command and Protocol Support, 360-368****SMTP connections, 848, 873****smtp_cr channel keyword, 363****smtp_crlf channel keyword, 363****smtp_crnlf channel keyword, 363****SMTP errors, os_smtp_* errors, 865****smtp_lf channel keyword, 363****SMTP MAIL TO command, 365****SMTP Proxy, 734, 746-749****MMP, 178****SMTP Relaying, adding, 557-559****SMTP Relaying for External Sites, allowing in NMS, 559****SMTP relays, 521****SMTP server slowdown, 848****SMTP service****access control, 541****adding relaying, 557-559****authenticated SMTP, 720****login requirements, 720****password-based login, 720****port number, 721****relay blocking, 560****starting and stopping, 124-127****SNMP, 891****applTable, 903****applTable Usage, 905****assocTable, 905-906****assocTable Usage, 906****channel errors, 909****channel information, 907****channel network connection, 909****configuring for Messaging Server, 893-894****HA, 898****implementation, 891-893****information provided, 903-910**

-
- SNMP (*Continued*)
 - limitations, 892
 - MIBs supported, 891
 - monitoring multiple instances, 897-898
 - MTA information, 906
 - mtaGroupAssociationTable, 908-909
 - mtaGroupErrorTable, 909-910
 - mtaGroupErrorTable Usage, 910
 - mtaGroupTable, 907-908
 - mtaGroupTable Usage, 908
 - mtaTable, 906-907
 - mtaTable Usage, 907
 - network connection information, 905
 - operation, 892-893
 - server information, 903
 - Standalone Agent, 897
 - subagent options, 899-902
 - snmp.listenaddr, 900
 - SOCKS_HOST, 498
 - SOCKS_PORT, 498
 - source channel-specific, rewriting, 310
 - source files, including, 236
 - source-routed address, 298
 - source routes, 399
 - sourceblocklimit, 411
 - sourcecommentinc, 396
 - sourcecommentmap, 396
 - sourcecommentomit, 396
 - sourcecommentstrip, 396
 - sourcecommenttota, 396
 - sourcefilter, 419, 572
 - sourcenosolicit, 424
 - sourcepersonalinc, 397
 - sourcepersonalmap, 397
 - sourcepersonalomit, 397
 - sourcepersonalstrip, 397
 - sourceroute, 390
 - sourcespamfilterXoptin, 420
 - sourcesrs, 519
 - Spam, *See* anti-spam
 - spam, *See* anti-spam, Brightmail and SpamAssassin
 - Spam filter, 570
 - Spam Filter options, 473-475
 - spamadjust, 504
 - SpamAssassin, 481
 - deploying, 483
 - examples, 483
 - file spam, 483-485
 - locating the server, 482-483
 - mode, 493
 - options (spamassassin.opt), 491-493
 - Requirements and Performance, 482-483
 - result, 481
 - score, 481
 - Theory of Operations, 481-482
 - verdict, 481
 - SpamAssassin, score, 488-489
 - spamd, 481
 - spamfilterX_action_n, 474
 - SpamfilterX_config_file, 473
 - spamfilterX_final, 475
 - SpamfilterX_library, 473
 - SpamfilterX_null_action, 474
 - SpamfilterX_null_optin, 474
 - SpamfilterX_optional, 473
 - SpamfilterX_string_action, 474
 - spamfilterX_verdict_n, 474, 485
 - spamtest, 504
 - special directives, 445
 - SPF, 511-520
 - spfquery, 516-518
 - SRS, 518-520
 - SSL
 - certificates, 722-731
 - ciphers, 732-733
 - enabling, 731-733
 - hardware encryption accelerators, 723
 - installing CA certificates, 726-731
 - internal and external modules, 722-723
 - optimizing performance, 734
 - overview, 720-734
 - password file for, 723-724
 - POP over, 145-146
 - sslpassword.conf file, 723-724
 - sslrootcacertsurl, 769
 - SSO, 157
 - configuring, 158-159
 - Cookie, 160

- SSO (*Continued*)
 limitations, 158
 Messenger Express configuration parameters, 158
 troubleshooting, 159-160
 trusted circle, 160-169
- SSR, 857
 syntax problems, 858
- standard procedures, MTA troubleshooting, 836
- start-msg, 125, 126
- starting individual channels, 840
- starting/stopping
 automatic server restart, 127-129
 HA servers, 124-125, 126, 127
 non-HA servers, 125-126
- starting/stopping servers, 124-127
- startup across reboots, 76
- status messages, *See* notification messages
- status notifications, *See* notification messages
- sticky error message, 313
- stop-msg, 125
- stopping inbound processing from a domain or IP address, 841
- stopping individual channels, 840
- stopping/startng servers, 124-127
- store.admins, 594-595
- store.cleanupage, 634
- store.defaultmailboxquota, 619, 620
- store.defaultmessagequota, 619
- store.defaultpartition, 637
- store.expirerule, 627
- store.quotaenforcement, 619, 623, 624
- store.quotaexceededmsg, 619, 621-623
- store.quotaexceededmsginterval, 619, 622
- store.quotagraceperiod, 619
- store.quotanotification, 619, 621-623
- store.quotawarn, 619, 622
- store_root, 591
- stored, 877
- stored operations, 663
- stored processes, troubleshooting the message store, 662
- streaming channel keyword, 368
- stripped Received, header lines, 853
- subaddresses, 398
- subaddressexact, 398
- subaddressrelaxed, 398
- subaddresswild, 398
- subdirs, 416
 how to use, 844
- subdirs channel keyword, 416
- submit channel keyword, 419
- substitutions, rewrite rules, unique string, 310
- substitutions in mapping templates, 244-245
- Sun Cluster, 87
- Sun ONE Console, 124
- sunManagedOrganization, 207
- sunPreferredDomain, 207
- SunPreferredDomain, 210
- suppressfinal, 279, 283
- swap space
 commands, 863
 errors, 863
- switchchannel, 394, 560
- switchchannel channel keyword, 374
- Symantec Anti-Virus Scanning Engine, *See* SASVE
- syntax checks after rewriting, 301
- syntax problems, SSR, 858

T

- tagged rewrite rule sets, 294
- tailor file, 255
- TCP client access control
 address-spoofing detection, 744
 examples, 743-744
 EXCEPT operator, 741-742
 filter syntax, 739-743
 host specification, 742
 how access filters work, 738
 identd service, 742-743
 overview, 737-746
 username lookup, 742-743
 virtual domains, 744
 wildcard names, 740-741
 wildcard patterns, 741

- TCP/IP
 channels, 253, 360
 connections, 368

TCP/IP (*Continued*)
 IDENT lookups, 372
 interface address, 371
 MX record support, 373
 port number, 370-371, 371
 reverse DNS lookups, 371
 TCP/IP channels, 359
 TCP/IP nameserver lookups, 373
`tcp_smtp_server` process, 523
 telemetry, 661-662
 TEXT_CHARSET, 277
 text database, 265-266
 threaddepth, 387
 threaddepth channel keyword, 381
 threads per process, 144
 throttle, 555
 timestampdelta, 769
 TLS, 145-146, 379
 channel keywords, 379
 description, 720
 tls channel keywords, 733
 TLS Problems, 846
 tlsswitchchannel keyword, 379
 traffic for conversion processing, 434-435
 transactionlimit, 386
 Transport Layer Security (TLS), 720
 trimming message header lines, 402
 troubleshooting
 login failure, POP, 140
 message store, 671-675
 slow email send, 858
 wildcards &, 672
 troubleshooting the message store, 660, 661
 common problems and solutions
 user mailbox directory problems, 673
 core files, 663
 database log files, 663
 hardware space, 661
 monitoring, 661
 stored operations, 663
 stored processes, 662
 user folders, 663
 troubleshooting the MTA
 checking configuration, 836

troubleshooting the MTA (*Continued*)
 checking the message queue directories, 836
 common problems
 changes to configuration files, 847
 looping messages, 853
 messages are not dequeued, 849
 messages not delivered, 852
 MTA does not receive incoming mail, 847
 received message is encoded, 857
 server-side rules, 857
 timeouts on SMTP connections, 848
 example, 842
 general error messages, 859
 file open or create errors, 864
 illegal host/domain errors, 864
 mm_init, 859
 os_smtp_* errors, 865
 swap space, 863
 version mismatch, 863
 .HELD messages, 854-856
 how to manually run a channel program, 839
 how to stop and start individual channels, 840, 843
 how to stop inbound processing from a domain or IP
 address, 841
 identify channels in message path, 842
 identifying the point of message breakdown, 845
 imsimta qm start, 840
 imsimta qm stop, 840
 imsimta test -rewrite, 836
 Job Controller and Dispatcher, 837
 log files, 838
 overview, 835
 ownership of files, 837
 standard procedures, 836
 truncatesmtplonglines, 414
 Trusted applications, 160
 Trusted circle, 160
 trustedurl, 769
 ttl parameter, 696
 two-digit dates, 402
 two-digit years, 402

U

UID, valid characters, 592-593
Unauthorized Bulk Email, 563-566
unconfiguring high availability, 119
undelivered messages, 383-384
uninstallation, high availability, 119
uniqueMember, 227
UNIX system users and groups, 63-64
unrecognized

 domain specification, 313
 host specification, 313

unrestricted, 395

unrestricted channel keyword, 395
unsolicited bulk email, *See* anti-spam
UpdateMsg parameter, 698-699
upgrade, 85

 migrating mailboxes, 675-686

upgrading from 5.2, 85

urgentbackoff, 383

urgentbackoff channel keyword, 381

urgentblocklimit, 387

urgentblocklimit channel keyword, 381

urgentnotices, 278

urgentnotices channel keyword, 382

USE_CHECK, 492

USE_DOMAIN_DATABASE, enabling direct
 LDAP, 231-232

USE_FORWARD_DATABASE, 269, 271

USE_REVERSE_DATABASE, 228-230, 266, 267, 270
 enabling direct LDAP, 231-232

use_text_databases, 250

useconfig utility, 95

useintermediate, 283

user

 access monitoring, 659-660

 removing, 123

user directory, 134-135

user folders, troubleshooting the message store, 663

User ID, valid characters, 592-593

user login., *See* login

user mailbox directory problems, troubleshooting the
 message store, 673

User Management Utilities, *See* Delegated
 Administrator

usercertfilter, 770

users, creation, 122

users and groups, UNIX system, 63-64

userswitchchannel, 375

using native sendmail configuration file, 77-78

utilities, 878-889

uucp, 390

UUCP address rewrite rules, 293

V

vacation caching, 408

VACATION_CLEANUP, 537

vacation messages, 533

vacation messages, forwarded email, 539-540

VACATION_TEMPLATE, 536, 537

vacationEndDate, 538

vacationStartDate, 537

Vanity domain, 206

vanity domains, 231-232

vdmap (Messaging Multiplexor), 177

verbosity (of logging), 821-822

verdict, 492, 498

VerifySSO, 168

verifyurl, 168

Veritas Cluster Server, 87, 116

 configuration, 116

version mismatch, 863

vertical bar (\\|), 294

viaaliasoptional, 400

viaaliasrequired, 400

virtual domains, controlling access to, 744

virus filtering, 463

virus scanning, 431

VRFY command, 365

VRFY Command Support, 365

vrfyallow channel keyword, 365

vrfydefault channel keyword, 365

vrfyhide channel keyword, 365

W

warnpost, 280

watcher, 127, 886-889
webmail
 HTTP service, 151-155
 Messenger Express, 137
wild cards, 672
wildcard characters, in mapping, 241
wildcardfield substitutions, 246
worksheet, 991
 comm_dssetup.pl, 993
 Directory Server, 991
 Messaging Server, 65, 994
wrapsmtplonglines, 414

X

x_env_to, 402
X-Envelope-to
 header lines
 generating, 402
X-REWRITE-SMS-ADDRESS mapping table, 929
XADR, 136
XCIR, 136
XGEN, 136
XSTA, 136

